新编应用型系列技能丛书

Android

应用程序设计

王英强　陈绥阳　张文胜 ● 主编

- 知识讲解“由浅入深、结构清晰”
- 教学方法采用“案例驱动+综合实训”
- 以培养学生的“工程应用能力”为目标
- 提供“立体化教材”

配套教学资源下载：www.tup.com.cn

清华大学出版社

新编应用型系列技能丛书

Android 应用程序设计

王英强　陈绥阳　张文胜　主编

清华大学出版社

北　京

内 容 简 介

本书介绍了基于 Android 操作系统的应用程序开发，内容由浅入深，讲述了在 Android 应用程序开发过程中最常用的一些技术。本书以学生为主体，理论联系实际，每一个章节除了讲述知识点外，都配有相应实例供学生实践，从而提高学生的动手实践能力。本书主要内容包括 Android 环境的搭建、布局管理、常用控件、菜单与消息提示、程序调试、数据存储、网络通信与服务、手机通信与设置和 Android 游戏制作等。

本书可作为普通高等学校的教材，也可作为高职高专院校的 Android 程序设计教材。

图书在版编目（CIP）数据

Android 应用程序设计/王英强，陈绥阳，张文胜主编. —北京：清华大学出版社，2013
（新编应用型系列技能丛书）

ISBN 978-7-302-33665-5

I. ①A…　II. ①王…　②陈…　③张…　III. ①移动终端-应用程序-程序设计　IV. ①TN929.53

中国版本图书馆 CIP 数据核字（2013）第 206353 号

责任编辑：苏明芳
封面设计：刘　超
版式设计：文森时代
责任校对：赵丽杰
责任印制：
出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
　　社 总 机：010-62770175　　**邮　　购：**010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　　张：**20　**字　　数：**468 千字
版　　次：2013 年 10 月第 1 版　**印　　次：**2013 年 10 月第 1 次印刷
印　　数：1～2400
定　　价：38.00 元

产品编号：054584-01

编审委员会

序
Preface

2012 年 12 月，清华大学出版社在西安组织了两次电子和计算机类的教材讨论会，着重评论了应用型本科与高职高专的教材编写问题。本系列丛书，即是针对应用型本科的一组教材。

面向应用型本科的教师和学生，提供一组电子和计算机类的教材，不仅是市场细分的要求，而且是应用型本科培养目标与培养模式的要求。在以学历文凭为目的的教育中，以应试为目标，其教学是以知识点为主，而不强调应用。高职高专的专业强调其社会属性，是面向工作岗位的，采用“校企合作、工学结合、顶岗实习”的培养模式，提倡情景教学与面向工作任务的工作过程式教学方法，其教材是技能导向的。“应用型本科”，一方面是有别于高职类型的普通本科，其专业既有学科属性，表现为知识的基础性、系统性、完整性和时序性，又有社会分工的社会属性；另一方面，“应用型”则表示该专业的人才培养目标，是为地区经济建设与社会发展服务的。兼顾这两方面的要求，这是本组教材应体现的基本特征。

专业的培养方案是规范培养过程以达到培养目标的基本文件，其中的教学计划与课程大纲（标准）则是建立了教学的课程体系与教学的内容体系。这一体系的建立是复杂而细致的工作，通常不是个人能力所及的事情。目前，市场上现有的教材过于强调教材自身的封闭性，造成教材内容的过度冗余与教材间内容的过度重复，往往也是这一原因而局限于一本教材自身所造成的。参加本组教材的院校，在多年的专业举办实践中，为适应市场的需求，不断进行课程体系与内容体系的建设工作，同时进行内容的梳理，在本组教材的编写过程中采用集体讨论、集体编写的方法，有助于体现相应的建设成果。

本系列教材中，有的课程是相关院校的精品课程、重点课程建设项目，在教学方法上有所探究，也有经验与教训。应当看到，精品课程、重点课程建设首先是针对教师的，是为教师提供一个可供示范的样本和教学资料。但教材，首先是面向学生的，本系列教材是面向应用型本科的学生，这有明确的定位。在这一定位上，有的教材在写作风格上追求“让学生能读懂”，经验说明，这是比较难做到的。事实上，在一本教材中，既有教师指导学生阅读的部分，又有学生自学的部分，这体现了教学过程中教师的主导作用，也是基于建构主义学习理论的。教师编写并出版的仅仅是文本，只有经过教师的使用和学生的阅读，才成为教材。

电子和计算机行业是科技进步很快的行业，反映到教材上是要兼顾基础性、工具性和现代性。在该系列教材中，有从基于 Windows 系统的程序架构到基于 Android 系统的应用程序设计，有从单片机接口技术与应用实践到嵌入式系统，有从语言类教材到基于数据库

的.Net 架构，既反映了基于 PC（Personal Computer，个人计算机）体系结构与互联网结构的应用软件系统设计，又反映了基于非 PC 体系结构的先进计算机系统与移动互联网的应用软件系统设计，既反映了 Java 中的 J2EE 技术，又反映微软 C#的技术，以适合 B/S 结构而满足中小型企业信息化建设的市场需求，读者有较大选择空间。

值得重视的是，在教材的编写中，编者引入了“计算思维”（《计算思维》，卡内基梅隆大学计算机系主任周以真）的观点，即不将计算机科学局限于编写代码，而着重于计算思维的培养。同时，在写作过程中，在涵盖知识点（用知识结构图表述）的基础上，采用基于任务的讲述过程，重视四个层次的实验，即认知性实验、验证性实验、设计性实验和综合性实验，并通过综合实例，以培养学生的应用能力。同时，积极进行立体化教材：含大纲，教学用 PPT、习题、习题答案、模拟试卷、模拟试卷答案、实训指导书等。

在本丛书的成书过程中，编者参考了多本相关书籍，作为附录加以注明，同时又得到清华大学出版社的大力支持，尤其是编辑苏明芳在编辑出版等方面做了大量工作，在此一并感谢。

由于编者学识有限，书中难免挂一漏万，存在不妥之处，敬请读者斧正。

Note

前 言
Foreword

随着我国 3G 网络的发展，智能手机也逐渐进入人们的日常生活。智能手机之所以能受到人们的欢迎，在于其高速的网络宽带、强大的功能以及随心所欲的个性化设置。在诸多的移动平台中，Android 是基于 Linux 平台开源的手机操作系统，是由 Google 公司和开放手机联盟共同开发的，以其优越的性能及开放性，受到了各手机厂商与通信运营商的推崇，迅速地占领了很大的市场份额。

本书从教学实际需求出发，合理安排知识结构，由浅入深，循序渐进，以应用为主，目的是提高学生的动手实践能力，缩小高等学校在人才培养上和软件公司在人才需求上的差距。

本书具有以下特色：

- 讲述由浅入深，从Android的基础知识到实际开发应用，结构清晰。本书以学生为主体，理论联系实际，每一个章节除了讲述知识点外，都配有相应实例供学生实践，从而提高学生的动手实践能力。
- 本书面向高等学校，目标是培养学生的工程应用能力，在教学方法上采用案例驱动与综合实训相结合的方式，本书的写作特点是基于任务的认知过程，由实例程序得到基本知识点，再进行知识拓展，并以学生实际动手写程序来完成一个知识单元的学习。最后一章是一个综合实训，将分散知识点的小实例综合为实训，有利于学生把知识点贯穿起来，形成系统性、完整性的项目体系。
- 提供立体化教材，提供下载教学用课件PPT、课程案例源代码等，方便学生学习。

本书共有 12 章，主要内容及各章节要求如下。

第 1 章　Android 概述：要求了解 Android 平台的发展历史。

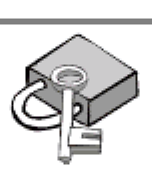
第 2 章　Android 开发平台的搭建与设置：要求了解创建 Android 程序的方法，掌握 Android 开发平台的搭建、Android 应用程序构成。

第 3 章　Activity 组件：要求了解 Activity 的生命周期，掌握 Activity 之间的调用及数据传送。

第 4 章　Android 布局管理：要求掌握 Android 中线性布局、相对布局、表格布局、帧布局、绝对布局的使用，了解布局之间的嵌套。

第 5 章　常用基本控件：要求掌握 TextView、EditText、Button、RadioButton、CheckBox 等基本控件的使用。

第 6 章　高级控件：要求掌握 AutoCompleteTextView、Spinner、ListView、GridView、ProgressBar、Gallery 等高级控件的使用。





第 7 章 菜单与消息提示：要求掌握选项菜单、上下文菜单、Alert 对话框、Toast、Notification 的使用方法。

第 8 章 Android 程序调试：要求掌握 Android 程序的调试方法、DDMS 的使用。

第 9 章 Android 数据存储与处理：掌握首选项、文件、数据库的访问方法，以及 Content Provider 类的使用方法。

第 10 章 网络通信与服务：掌握消息广播、Service 的使用，了解 HTTP 网络通信、WebView 控件、E-mail 的发送。

第 11 章 手机通信与设置：掌握拨打电话、收发短信的方法，了解手机声音与手机闹钟的设置方法。

第 12 章 Android 游戏制作：为了提升读者对 Android 的学习，本章介绍了一个综合实例，从项目的系统需求分析开始，然后进行系统设计和模块划分，最后进行代码的设计，让读者熟悉一个项目完整的开发过程。

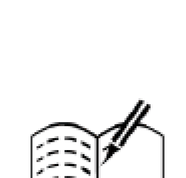
在学时设计上，总量控制为 94 学时，其中 64 学时为教学时数，可分为教学 48 学时、实验 16 学时（或教学 40 学时、实验 24 学时），本书按 64 学时进行内容选取，另有 30 学时的综合实训，其源程序代码通过立体化教材在网站上提供，不在本书内反映。

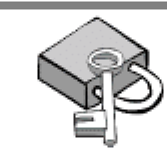
本书由王英强、陈绥阳、张文胜主编。第 1～11 章由王英强编写，第 12 章由张文胜编写，由陈绥阳教授统稿并审稿。此外，在编写本书的过程中，很多同事给予了很大的帮助，其中王征风、王红刚、王振铎等为本书实例的编写提供了大量的素材，清华大学出版社的苏明芳老师也提出了很多意见，为本书的出版付出了很多努力。在此，编者对他们表示衷心的感谢。由于编者水平有限，本书难免有不足之处，欢迎广大读者批评指正。读者对本书有任何建议，可发送 E-mail 至 y_q_wang@163.com。

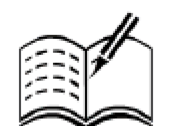
编　者

目　录

Contents


Note




Note

第 1 章 Android 概述

【本章内容】

- ❑ Android简介
- ❑ Android发展历史
- ❑ Android平台框架
- ❑ Android基本组件

Android 是一款以 Linux 为基础的开放源代码的操作系统，主要使用于便携设备，是由谷歌（Google）与开放手机联盟（Open Handset Alliance）共同提供的软件平台，有望为全球手机市场带来革命性的变化。2011 年第一季度，Android 在全球的市场份额首次超过塞班系统，跃居全球第一。2011 年 11 月数据显示，Android 占据全球智能手机操作系统市场 52.5%的份额，中国市场占有率为 58%。随着 Android 手机的普及，Android 应用的需求势必会越来越大，这将是一个潜力巨大的市场，吸引着广大的软件开发厂商和开发者投身其中。

1.1 Android 简介

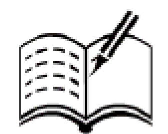
Android 一词来源于法国作家维里耶德利尔·亚当在 1886 年发表的科幻小说《未来夏娃》，本意是“机器人”。虽然 Android 平台是由 Google 公司推出的，但更确切地说应该是开放手机联盟的产品。开放手机联盟是由 30 多家高科技公司和手机公司组成的，包括 Google、HTC（宏达电子）、T-Mobile、高通、摩托罗拉、三星、LG 以及中国移动等。开放手机联盟表示，Android 是本着成为第一个开放、完整、免费、专门针对移动设备开发平台这一目标，完全从零开始创建的，因此 Android 是第一个完整、开放、免费的手机平台。

Android 系统具有以下特点：

（1）开放性。Google 通过与运营商、设备制造商、开发商等结成深层次的合作伙伴，通过建立标准化、开放式的移动电话软件平台，形成一个开放式的产业系统。

（2）平等性。在 Android 平台上，系统提供的软件和个人开发的应用程序是平等的。例如自己开发的拨打电话程序可以替代系统提供的相应程序。

（3）应用程序之间的沟通很方便。在 Android 平台下开发的应用程序，可以很方便地


Note

实现应用程序之间数据的共享，只需要进行简单的声明和操作，应用程序就可以访问或者调用其他应用程序的数据，或者将自己的数据提供给其他应用程序使用。

1.2　Android 发展历史

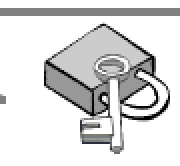
2005 年，Google 收购了仅成立 22 个月的高科技企业 Android，2007 年，正式向外界展示了 Android 操作系统，2008 年 9 月 23 日，Google 发布 Android 1.0，从此就有了今天风靡全球的 Android。

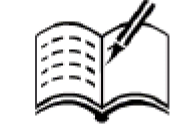
自 Android 1.5 开始，Android 用甜点作为系统版本的代号，随着版本的升级，代表的甜点的尺寸越变越大，并按照字母顺序排列，依次为纸杯蛋糕、甜甜圈、松饼、冻酸奶、姜饼、蜂巢、冰激凌三明治等。

Android 发行的各版本及其特征如表 1-1 所示。

表 1-1　Android 发行版本及其特征

版　本	特　征
Android 1.1	2008 年 9 月发布的 Android 第 1 版
Android 1.5	2009 年 4 月 30 日，官方 1.5 版本（Cupcake，纸杯蛋糕）的 Android 发布。主要更新如下： 1．拍摄/播放影片，并支持上传到 YouTube 2．支持立体声蓝牙耳机，同时改善自动配对性能 3．采用 WebKit 技术的浏览器，支持复制/粘贴和页面中搜索 4．GPS 性能大大提高 5．提供屏幕虚拟键盘 6．主屏幕增加音乐播放器和相框 Widget 7．应用程序自动随着手机旋转 8．短信、Gmail、日历，浏览器的用户接口大幅改进，如 Gmail 可以批量删除邮件 9．相机启动速度加快，拍摄图片可以直接上传到 Picasa 10．来电照片显示
Android 1.6	2009 年 9 月 15 日，1.6 版本（Donut，甜甜圈）软件开发工具包发布。主要更新如下： 1．重新设计的 Android Market 手势 2．支持 CDMA 网络 3．文字转语音系统（Text-to-Speech） 4．快速搜索框 5．全新的拍照接口 6．查看应用程序耗电 7．支持虚拟私人网络（VPN） 8．支持更多的屏幕分辨率 9．支持 OpenCore 2 媒体引擎 10．新增面向视觉或听觉困难人群的易用性插件





续表

版　本	特　征
Android 2.0/2.0.1/2.1	2009 年 10 月 26 日，2.0 版本（Éclair，松饼）软件开发工具包发布。主要更新如下： 1．优化硬件速度 2．Car Home 程序 3．支持更多的屏幕分辨率 4．改良的用户界面 5．新的浏览器的用户接口和支持 HTML 5 6．新的联系人名单 7．更好的白色/黑色背景比率 8．改进 Google Maps 3.1.2 9．支持 Microsoft Exchange 10．支持内置相机闪光灯 11．支持数码变焦 12．改进的虚拟键盘 13．支持蓝牙 2.1 14．支持动态桌面的设计
Android 2.2/2.2.1	2010 年 5 月 20 日，2.2 版本（Froyo，冻酸奶）软件开发工具包发布。主要更新如下： 1．整体性能大幅度地提升 2．3G 网络共享功能 3．支持 Flash 4．App 2 SD 功能 5．全新的软件商店 6．更多的 Web 应用 API 接口的开发
Android 2.3	2010 年 12 月 7 日，2.3 版本（Gingerbread，姜饼）软件开发工具包发布。主要更新如下： 1．增加了新的垃圾回收和优化处理事件 2．原生代码可直接存取输入和感应器事件、EGL/OpenGL ES、OpenSL ES 3．新的管理窗口和生命周期的框架 4．支持 VP8 和 WebM 视频格式，提供 AAC 和 AMR 宽频编码，提供了新的音频效果器 5．支持前置摄像头、SIP/VOIP 和 NFC（近场通信） 6．简化界面、提升速度 7．更快、更直观的文字输入 8．一键文字选择和复制/粘贴 9．改进的电源管理系统 10．新的应用管理方式
Android 3.0	2011 年 2 月 2 日，3.0 版本（Honeycomb，蜂巢）发布。主要更新如下： 1．针对平板进行优化 2．全新设计的 UI 增强网页浏览功能 3．In-App Purchase 功能

Note

续表

版　本	特　征
Android 4.0	2011 年 10 月 19 日，4.0 版本（Ice Cream Sandwich，冰激凌三明治）在香港发布。主要更新如下： 1．全新的 UI 2．全新的 Chrome Lite 浏览器，有离线阅读、16 标签页、隐身浏览模式等 3．截图功能 4．更强大的图片编辑功能 5．自带照片应用堪比 Instagram，可以加滤镜、相框，进行 360°全景拍摄，照片还能根据地点来排序 6．Gmail 加入手势、离线搜索功能，UI 更强大 7．新功能 People：以联系人照片为核心，界面偏重滑动而非单击，集成了 Twitter、Linkedin、Google+等通信工具。有望支持用户自定义添加第三方服务 8．新增流量管理工具，可具体查看每个应用产生的流量 9．正在运行的程序可以互相切换 10．人脸识别功能 11．系统优化，速度更快 12．支持虚拟按键，手机可以不再拥有任何按键 13．更直观的程序文件夹 14．平板电脑和智能手机通用 15．支持更高的分辨率 16．专为双核处理器编写的优化驱动 17．全新的 Linux 内核 18．增强的复制、粘贴功能 19．语音功能 20．全新通知栏 21．更加丰富的数据传输功能 22．更多的感应器支持 23．语音识别的键盘 24．全新的 3D 驱动，游戏支持能力提升 25．全新的谷歌电子市场 26．增强桌面插件的自定义功能
Android 4.1/4.2	Jelly Bean（果冻豆），继“冰激凌三明治”之后的下一版 Android 系统

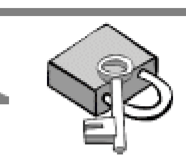
注：本书所有实例在模拟器上进行测试，完全兼容于 Android SDK 中 Android 2.1 以上版本。

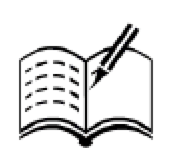
1.3　Android 平台框架

在 1.2 节介绍了 Android 平台的发展历史及其特征，本节将对 Android 内部的系统框架进行介绍。Android 平台框架如图 1-1 所示。Android 平台框架中的各组成部分介绍如下。

1．Linux Kernel（Linux 内核）

Android 基于 Linux 2.6 提供核心系统服务，如安全、内存管理、进程管理、网络堆栈、





驱动模型。Linux Kernel 也作为硬件和软件之间的抽象层，隐藏具体硬件细节而为上层提供统一的服务。如果只是进行应用程序开发，则不需要深入了解 Linux Kernel 层。

2．Libraries（库）

Android 包含一个 C/C++库的集合，供 Android 系统的各个组件使用。这些功能通过 Android 的应用程序框架（Application Framework）（如图 1-1 所示）展现给开发者。下面列出一些核心库。

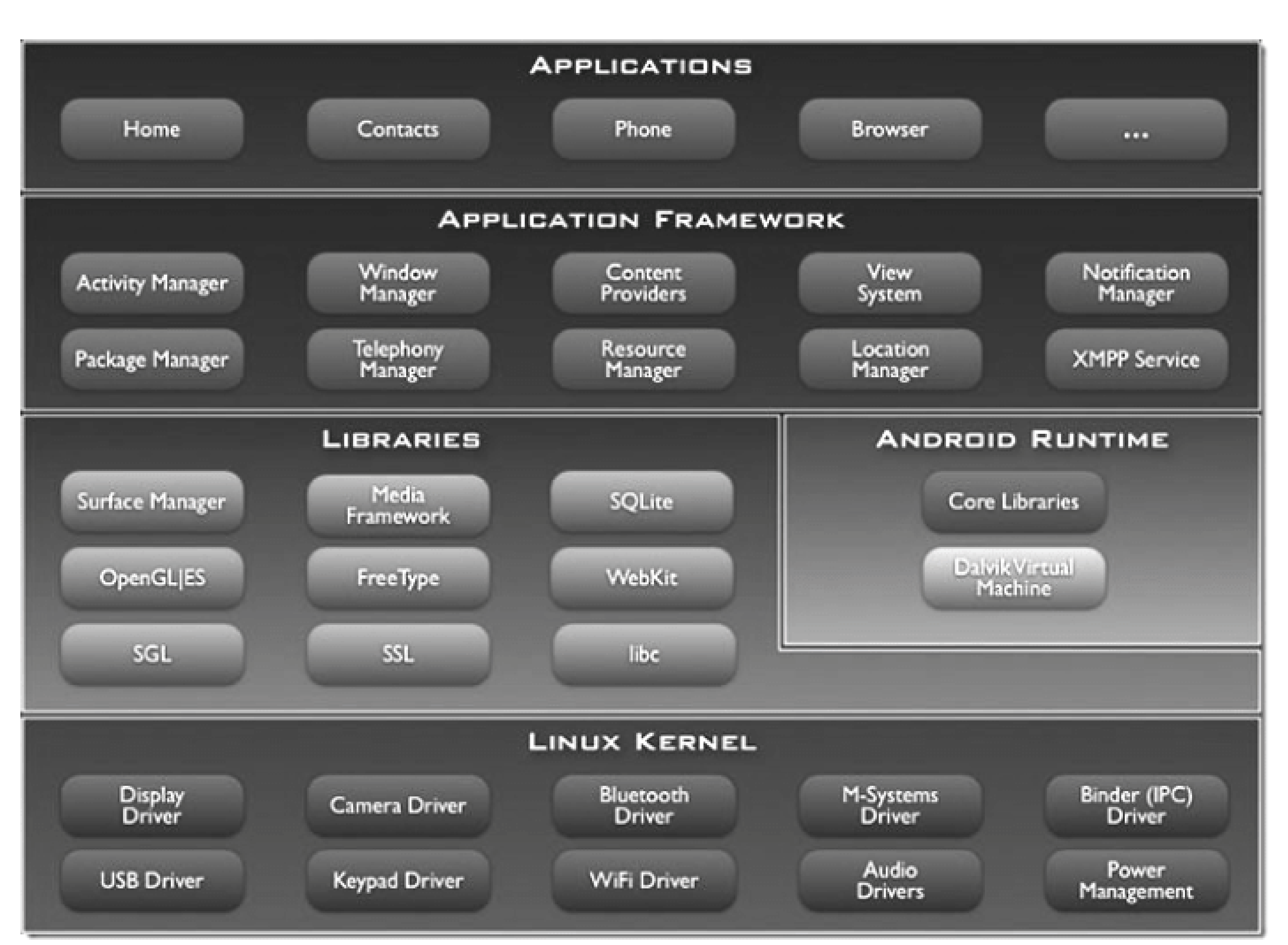


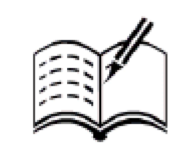
图 1-1　Android 平台应用程序框架图

- Libc：标准C系统库的BSD衍生，并为基于嵌入式Linux设备进行了优化。
- Media Framework：基于PacketVideo的OpenCore，该库支持播放和录制许多流行的音频和视频格式，以及静态图像文件，包括MPEG4、H.264、MP3、AAC、AMR、JPG、PNG等。
- Surface Manager：管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层。
- WebKit：新式的Web浏览器引擎，驱动Android 浏览器和内嵌的Web视图。
- SGL：基本的2D图形引擎。
- OpenGL|ES：基于OpenGL|ES 1.0 APIs实现，使用硬件3D加速，包含高度优化的3D软件光栅。
- FreeType：位图和矢量字体渲染。
- SQLite：所有应用程序都可以使用的强大而轻量级的关系数据库引擎。
- SSL：为网络通信提供安全及数据完整性的一种安全协议。

3．Android Runtime（Android 运行时）

Android 是包含一个核心库的集合，提供大部分在 Java 编程语言核心类库中可用的功能。每一个 Android 应用程序是 Dalvik 虚拟机中的实例，运行在它们自己的进程中。Dalvik

虚拟机依赖于 Linux 内核提供基本功能，来实现进程、内存和文件系统管理等各种服务，可以在一个设备中高效地运行多个虚拟机，可执行文件格式是.dex。.dex 格式是专为 Dalvik 设计的一种压缩格式，占用内存非常小，适合内存和处理器速度有限的系统。


Note

大多数虚拟机包括 JVM 都是基于栈的，而 Dalvik 虚拟机则是基于寄存器的。两种架构各有优劣，一般而言，基于栈的机器需要更多指令，而基于寄存器的机器指令更大。dx 是一套工具，可以将 Java 的.class 转换成.dex 格式。一个.dex 文件通常会有多个.class。由于.dex 有时必须进行优化，会使文件大小增加 1～4 倍。

4．Application Framework（应用程序框架）

通过提供开放的开发平台，Android 使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等。

应用程序的体系结构简化了组件之间的重用，任何应用程序服从框架执行的安全限制，都能发布自己的功能。通过应用程序框架，开发人员可以自由地使用核心应用程序所使用的框架 API 来实现自己程序的功能，替换系统应用程序。

所有的应用程序其实是一组服务和系统，包括以下内容。

- ❑ 视图提供者（View Providers）：丰富的、可扩展的视图集合，可用于构建一个应用程序，包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器。
- ❑ 内容提供者（Content Providers）：使应用程序能访问其他应用程序（如通讯录）的数据，或共享自己的数据。
- ❑ 资源管理器（Resource Manager）：提供访问非代码资源，如本地化字符串、图形和布局文件。
- ❑ 通知管理器（Notification Manager）：使所有的应用程序能够在状态栏显示自定义警告。
- ❑ 活动管理器（Activity Manager）：管理应用程序生命周期，提供通用的导航回退功能。

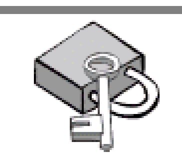
5．Application（应用程序）

Android 提供了一系列核心应用程序，包括电子邮件客户端、SMS 程序、拨打电话、日历、地图、浏览器、联系人和其他设置。这些应用程序都是用 Java 编程语言写的，而应用程序的开发人员可以开发出更多有创意、功能更强大的应用程序。

1.4　Android 基本组件

Android 的一个主要特点是，一个应用程序可以利用其他应用程序的元素（假设这些应用程序允许）。相反，当需求产生时它只是启动其他应用程序块。

对于这个工作，当应用程序的任何部分被请求时，系统必须能够启动一个应用程序的



进程，并实例化该部分的 Java 对象。因此，不像其他大多数系统的应用程序，Android 应用程序没有一个单一的入口点（例如，没有 main()函数）。相反，系统实例化和运行需要几个必要的组件，有 4 种类型的组件：活动（Activity）、服务（Service）、广播接收者（Broadcast Receiver）和内容提供者（Content Provider）。然而，并不是所有的应用程序都必须包含这 4 个部分，一个应用程序可以由其中的一个或几个来组建。下面介绍 Android 平台下的几个基本组件。

Note

1．活动（Activity）

Activity 是 Android 中最常用的组件，是应用程序的表示层，一般通过 View 来实现用户界面。一个活动表示一个可视化的用户界面，关注一个用户活动的事件。

一个应用程序可能只包含一个活动，也可能包含几个活动。这些活动是什么？有多少？这取决于它的应用和设计。虽然它们一起工作形成一个整体的用户界面，但是每个活动是独立于其他活动的，每一个都是作为 Activity 父类的一个子类。一般来讲，当应用程序被启动时，被标记为第一个的活动应该展示给用户，从一个活动移动到另一个活动由当前的活动完成。

窗口的可视内容是由继承自 View 父类的一个分层视图对象提供的，每个视图控件是窗口内的一个特定的矩形空间。一个视图是活动与用户交互发生的地方，例如，一个视图可能显示一个小的图片和当用户单击图片时发起一个行为。Android 有一些现成的视图可以使用，包括按钮（Button）、文本域（TextView、EditText）、复选框（CheckBox）和列表视图（ListView）等。

2．服务（Service）

一个服务没有一个可视化用户界面，而是在后台无期限地运行，例如，一个服务可能是播放背景音乐而用户做其他一些事情，或者从网络获取数据，或者计算一些东西并提供结果给需要的活动（Activity）。每个服务都继承自 Service 基类。

一个典型的例子是一个媒体播放器播放列表中的歌曲，该播放器应用程序将可能有一个或多个活动，允许用户选择歌曲和开始播放。然而，音乐播放本身不会被一个活动处理，因为当用户离开播放器去做其他事情时，仍希望保持音乐继续播放。为此，媒体播放器活动可以启动一个服务在后台运行，甚至当媒体播放器离开屏幕时，系统将保持音乐播放服务运行。

像活动和其他组件一样，运行在应用程序进程中的主线程中。因此，它们将不会阻止其他组件或用户界面，而是产生其他一些耗时的任务（如音乐播放）。

Service 从启动到销毁的过程会经历如下 3 个阶段：创建服务 onCreate()、开始服务 onStart()和销毁服务 onDestroy()。Service 的启动有两种方式：开始服务 context.startService()和绑定服务 context.bindService()。

（1）开始服务（startService）：在同一个应用的任何地方调用 startService()方法都能启动 Service，然后系统会回调 Service 类的 onCreate()和 onStart()方法。这样启动的 Service 会一直运行在后台，直到 Context.stopService()或者 selfStop()方法被调用。另外，如果一个

Service 已经被启动，其他代码再试图调用 startService()方法，是不会执行 onCreate()的，但会重新执行一次 onStart()。

（2）绑定服务（bindService）：把 Service 和调用该 Service 的客户类绑定起来，如果调用这个客户类被销毁，Service 也会被销毁。用这个方法的一个好处是，bindService()方法执行后，Service 会回调用上面提到的 onBind()方法，可以从这里返回一个实现了 IBind 接口的类，在客户端操作该类就能和服务通信了，如得到 Service 运行的状态或其他操作。如果 Service 还没有运行，使用这个方法启动 Service 就会调用 onCreate()方法而不会调用 onStart()方法。

Note

3．广播接收者（Broadcast Receiver）

一个广播接收者接收广播公告时可以做出相应的反应。许多广播源自于系统代码，如公告时区的改变、电池电量低、已采取图片、用户改变了语言偏好等。应用程序也可以发起广播，如为了让其他程序知道某些数据已经下载到设备且它们可以使用这些数据。

一个应用程序可以有任意数量的广播接收者去反映它认为重要的任何公告。所有的接收者继承自 BroadcastReceiver 基类。广播接收者不显示一个用户界面，然而，它们可以启动一个活动去响应收到的信息，或者使用 NotificationManager 通知用户。通知可以使用多种方式获得用户的注意，如闪烁的背光、振动设备、播放声音等。典型的方式是放置一个持久的图标在状态栏，用户可以打开获取信息。

4．内容提供者（Content Provider）

内容提供者可以将一个应用程序的指定数据集提供给其他应用程序，这些数据可以存储在文件系统中、SQLite 数据库或者任何其他合理的方式。内容提供者继承自 ContentProvider 父类并实现了一个标准的方法集合，使得其他应用程序可以检索和存储数据。

内容提供者是 Android 应用程序的主要组成部分之一，它们封装数据且通过 ContentResolver 接口提供给应用程序。只有需要在多个应用程序间共享数据时才使用内容提供者。例如，通讯录数据被多个应用程序使用，且必须存储在一个内容提供者中。如果不需要在多个应用程序间共享数据，可以直接使用 SQLite 数据库或者文件来保存数据。

1.5 习　　题

1．简述 Android 平台的特点。
2．简述 Android 平台框架的各组成部分及其作用。
3．简述 Android 的 4 个基本组件及其作用。
4．简述服务（Service）启动的两种方式。

第 2 章

Android 开发平台的搭建与设置

【本章内容】

- ❑ Android 开发工具介绍
- ❑ Android 平台搭建
- ❑ 创建 Hello Android 项目
- ❑ Android 应用程序构成

本章主要介绍 Android 开发平台的软硬件要求及搭建与配置，然后通过一个 Hello Android 项目向读者演示 Android 平台下应用程序的开发过程，并且对 Android 应用程序的构成进行介绍，主要目的是让读者了解 Android 平台的搭建及 Android 应用程序的构成。

2.1 Android 开发工具

进行 Android 应用程序开发，使用的工具主要有 JDK、Eclipse、Android SDK 及 Android 的支持插件 ADT，下面对上述工具进行介绍。

1. JDK

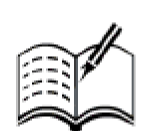
Android 平台下应用程序的开发主要采用 Java 语言。JDK（Java Development Kit）是 Sun Microsystems 针对 Java 开发员的产品。自从 Java 推出以来，JDK 已经成为使用最广泛的 Java SDK。JDK 是整个 Java 的核心，包括 Java 运行环境、Java 工具和 Java 基础的类库。Sun Microsystems 于 2009 年 4 月被 Oracle 公司收购，所以现在 JDK 可以从 Oracle 公司的官方网站上获取，打开浏览器，在地址栏输入地址 http://www.oracle.com/technetwork/java/javase/downloads/index.html，打开后的页面如图 2-1 所示。

2. Eclipse

Eclipse 是一种基于 Java 的可扩展开源开发平台。就其自身而言，它只是一个框架和一组服务，通过插件组件构建开发环境。另外，Eclipse 附带了一个标准的插件集，包括为人熟知的 Java 开发工具 JDT（Java Development Tools）。

虽然大多数用户很乐于将 Eclipse 当作 Java 集成开发环境（IDE）来使用，但 Eclipse 的目标却不仅限于此。Eclipse 还包括插件开发环境（Plug-in Development Environment，PDE），这个组件主要针对希望扩展 Eclipse 的软件开发人员，因为它允许构建与 Eclipse




Note

环境无缝集成的工具。由于 Eclipse 中的每样东西都是插件，对于给 Eclipse 提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥场所。



图 2-1　JDK 下载页面

这种平等和一致性并不仅限于 Java 开发工具。尽管 Eclipse 是使用 Java 语言开发的，但它的用途并不限于 Java 语言，例如，预计将会推出支持诸如 C/C++和 COBOL 等编程语言的插件，或已经可用。Eclipse 框架还可用来作为与软件开发无关的其他应用程序类型的基础，比如内容管理系统。

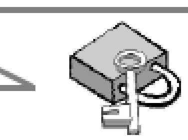
下面是目前已知的 Eclipse 版本代号：

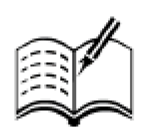
- Eclipse 3.1 版本代号 IO（木卫 1，伊奥）。
- Eclipse 3.2 版本代号 Callisto（木卫 4，卡里斯托）。
- Eclipse 3.3 版本代号 Europa（木卫 2，欧罗巴）。
- Eclipse 3.4 版本代号 Ganymede（木卫 3，盖尼米德）。
- Eclipse 3.5 版本代号 Galileo（伽利略）。
- Eclipse 3.6 版本代号 Helios（太阳神）。
- Eclipse 3.7 版本代号 Indigo（靛青）。

Eclipse 可以从 eclipse.org 网站（http://www.eclipse.org/downloads）下载，如图 2-2 所示。Eclipse 不需要安装，只需要将下载的压缩包解压到硬盘上的某个目录下即可。

3. Android SDK

Android SDK（Software Development Kit）是 Android 专属的软件开发工具包，可以从 Android 的官方网站上免费下载。在浏览器地址栏中输入 http://developer.android.com/sdk/index.html，打开如图 2-3 所示的页面，选择相应操作系统平台的压缩包，下载即可。





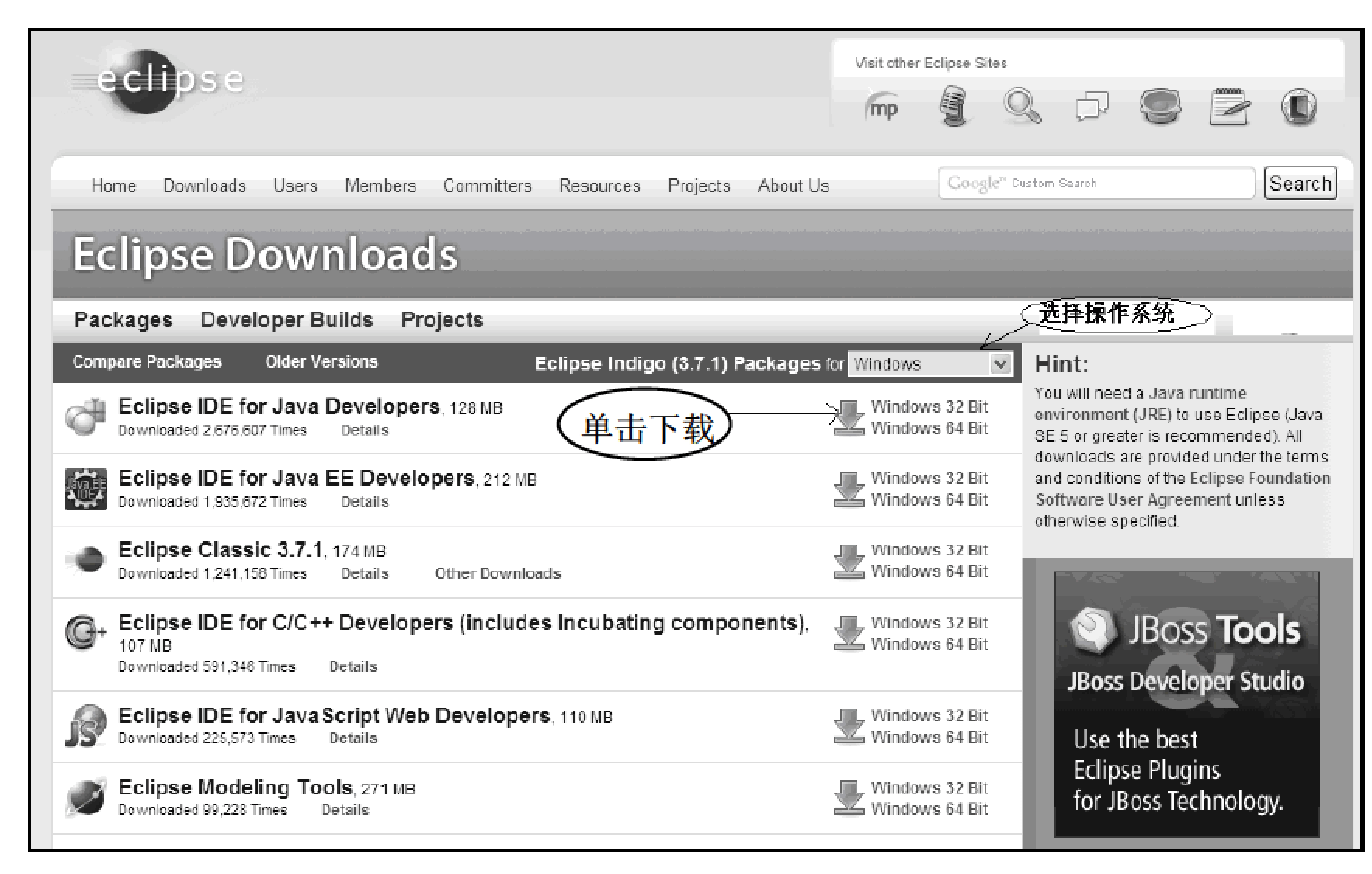


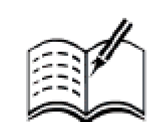
图 2-2　Eclipse 下载页面



图 2-3　Android SDK 下载页面

4.　ADT

ADT 是 Eclipse 平台下用来开发 Android 应用程序的插件。目前，Android 开发所用的开发工具是 Eclipse，为了使 Android 应用程序的创建、运行和调试更加方便，Android 开发团队专门针对 Eclipse IDE 定制该插件。ADT 可以在线安装，地址为 https://dl-ssl.google.com/android/eclipse/。


Note

2.2 搭建与设置 Android 开发平台

2.1 节中介绍了 Android 应用程序开发工具及其获取方法，按照上述方法获取到各开发工具的安装文件之后，就可以进行 Android 应用程序开发平台的搭建了。

1. 安装 JDK

双击并运行下载的 JDK 安装文件，根据安装提示，将 JDK 安装到指定位置，本书中将其安装在 D:\Program Files\Java\jdk1.6.0_20 目录下。

安装完毕后，检查系统的环境变量。方法为：右击“我的电脑”选择“属性”命令，选择“高级”选项卡，单击“环境变量”按钮，弹出“环境变量”对话框，如图 2-4 所示。增加 CLASSPATH 变量，值为 D:\Program Files\Java\jdk1.6.0_20\demo;D:\Program Files\Java\jdk1.6.0_20\lib；在 Path 变量的值后面增加 D:\Program Files\Java\jdk1.6.0_20\bin。

2. 安装 Eclipse

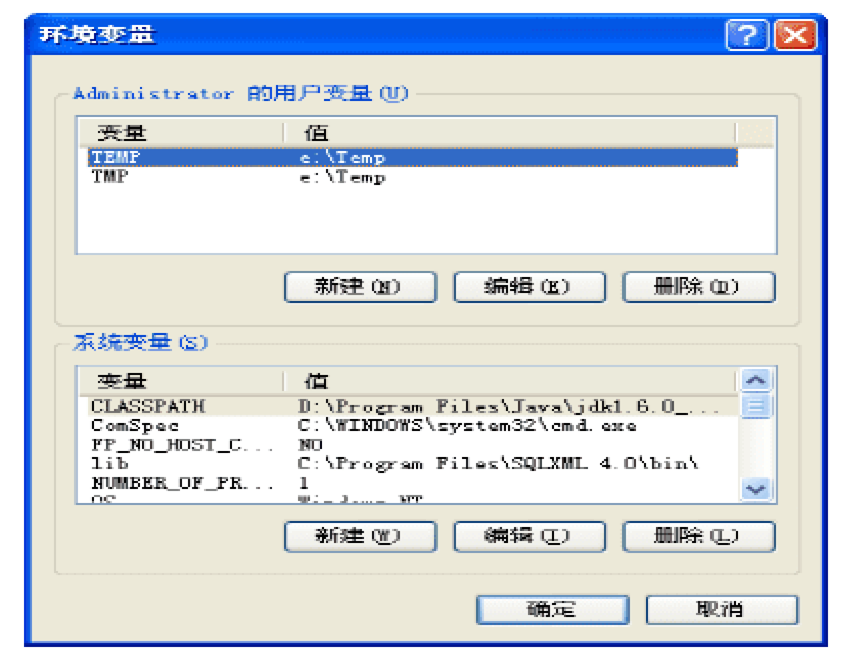
Eclipse 不需要安装，只需将下载的压缩包解压到硬盘上的某个目录下即可。本书中将其解压到 D:\Program Files\Eclipse 目录下。

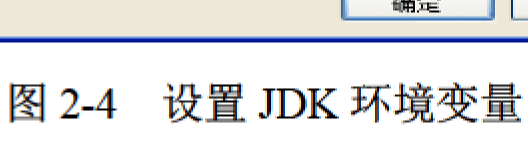
3. 安装 Android SDK

Android SDK 的安装过程比较简单，但是所花费的时间比较长，需要耐心等待。

（1）双击运行下载的 Android SDK，将其解压到硬盘上的某个位置。本书将其解压到 D:\Program Files\android-sdk-windows 目录下。

（2）运行 D:\Program Files\android-sdk-windows 下的 SDK Manager.exe，程序将会自动检测是否有新的 SDK 可以下载，检查结果如图 2-5 所示。




图 2-4 设置 JDK 环境变量

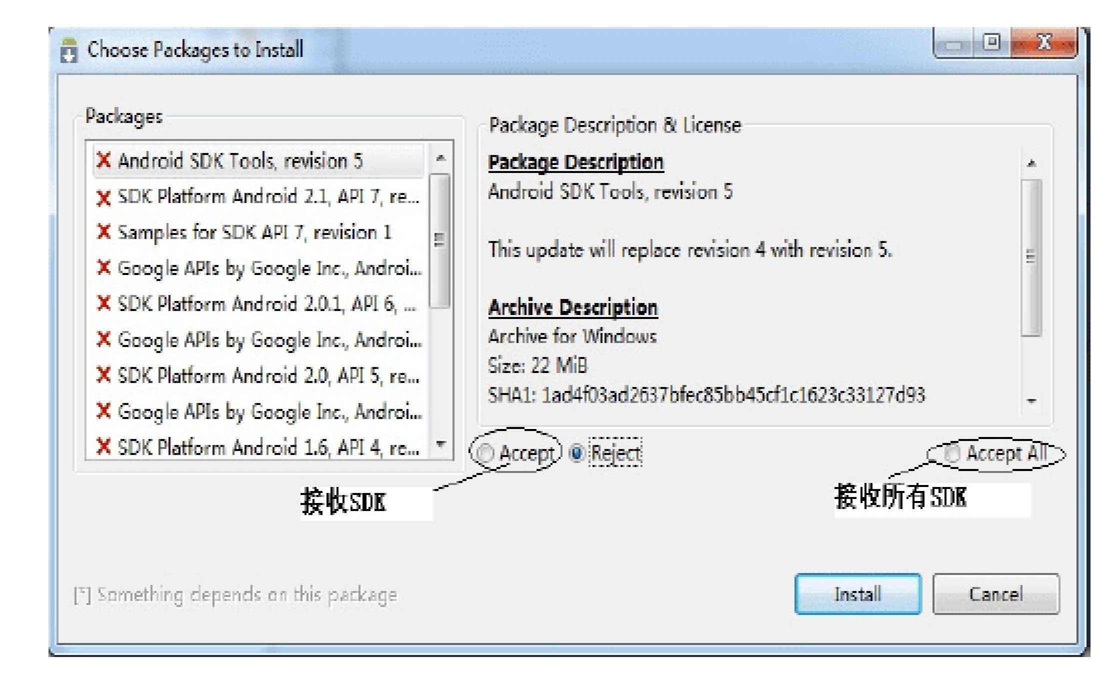
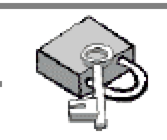


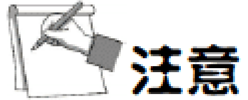
图 2-5 Android SDK 检查结果

（3）选择自己所需要的开发平台，例如 SDK Platform Android 2.0 指 Android 2.0 开发平台对应的 SDK。单击 Install 按钮进行安装。

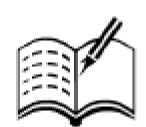
（4）右击“我的电脑”选择“属性”命令，选择“高级”选项卡，单击“环境变量”按钮，弹出“环境变量”对话框，在 Path 变量的值后面增加;D:\Program Files\android-



sdk-windows\tools，如图 2-6 所示。



注意



Note

安装 Android SDK 是一个漫长的过程，往往需要几个小时。为了减少等待时间，可以将所需要的 SDK 下载下来，解压到 Android 的 platforms 文件夹下，这样就不需要下载 SDK 进行安装，从而减少安装时间。安装结束之后文件列表如下。

- add-ons：一些扩展库，如 Google APIs Add-On。
- docs：API 文档等。
- platforms：各个版本的平台组件。
- samples：一些实例程序。
- tools：各种辅助工具。
- usb_driver：Windows 下的一些 USB 驱动。
- temp：存放下载平台组件过程中的临时文件。

4. 安装 ADT

ADT 的安装可以通过在线安装完成。

（1）运行 D:\Program Files\Eclipse 目录下的 Eclipse.exe，启动后，选择 Help 菜单下的 install New Software 命令，弹出如图 2-7 所示的界面。

（2）单击 Add 按钮，弹出 Add Site（添加新站点）的界面，即 Add Repository 对话框，如图 2-8 所示。在 Name 文本框中输入站点名称 ADT（该名字可以自行命名），在 Location 文本框中输入 http://dl-ssl.google.com/android/eclipse/，输入完成后单击 OK 按钮。



图 2-6　设置 SDK 环境变量

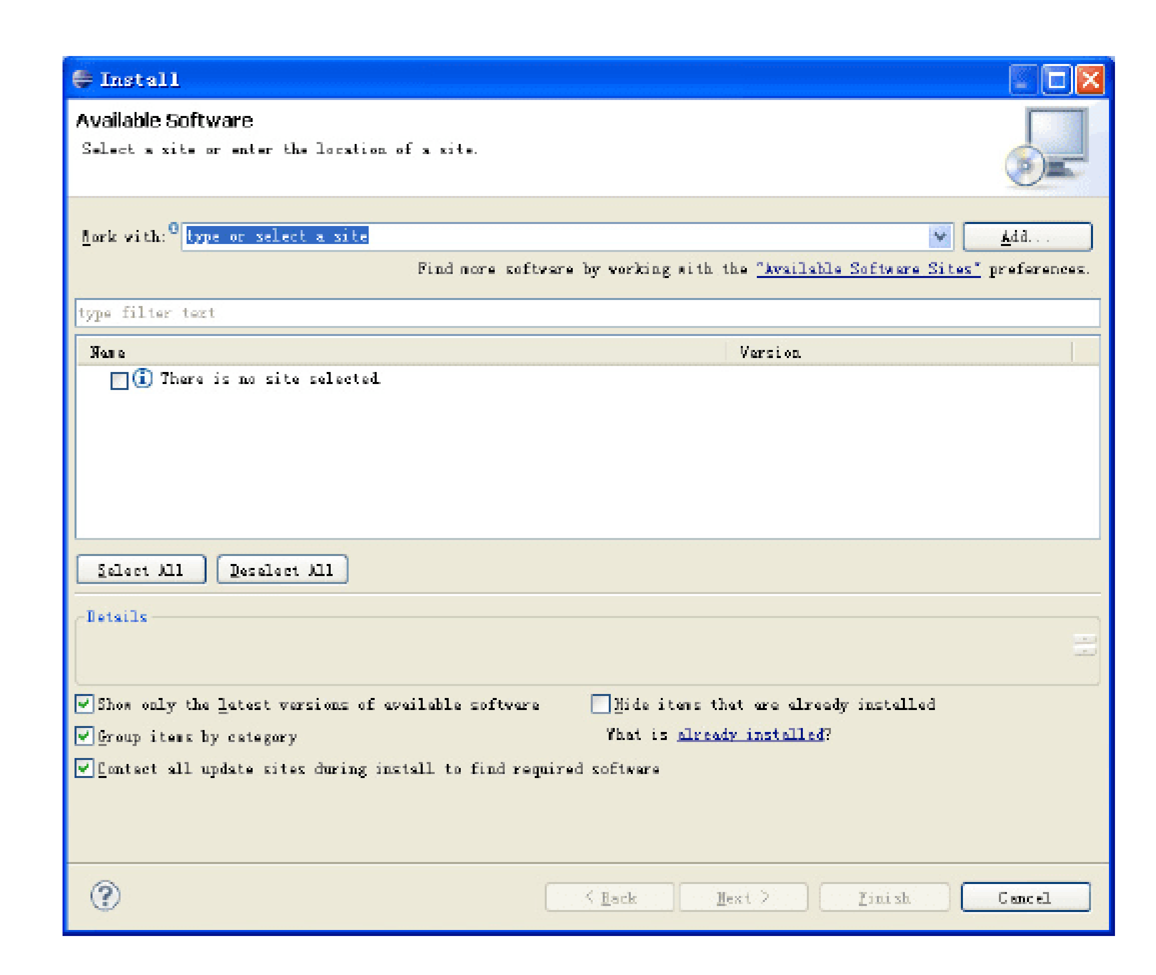


图 2-7　ADT 安装界面

（3）Eclipse 会自动连接到该站点，并将连接结果显示在列表中。在列表中会看到一个名为 Developer Tools 的选项，它包含两个子节点：Android DDMS 和 Android Development Tools。选择父节点 Developer Tools，并确认同时选中了两个子节点，然后单

击 Next 按钮，再根据安装向导进行安装，直到安装结束。



注意



DDMS 是 Android 应用程序开发一个很重要的调试工具，Android Development Tools 即 Android 开发工具。建议两个选项全部安装。

（4）在安装的过程中会弹出“插件中包含未注册内容”的安全警告，单击 OK 按钮继续安装。安装结束后会弹出是否重启 Eclipse 的提示框，单击 Yes 按钮重新启动 Eclipse。

（5）重新启动 Eclipse 之后，选择 Window/Preferences 命令，打开 Preferences 窗口，在左侧选择“Android”，在 SDK Location 文本框中输入 Android SDK 的安装路径。在本书中 Android SDK 安装在 D:\Program Files\android-sdk-windows 目录下，如图 2-9 所示，单击 OK 按钮，完成 ADT 插件的安装。

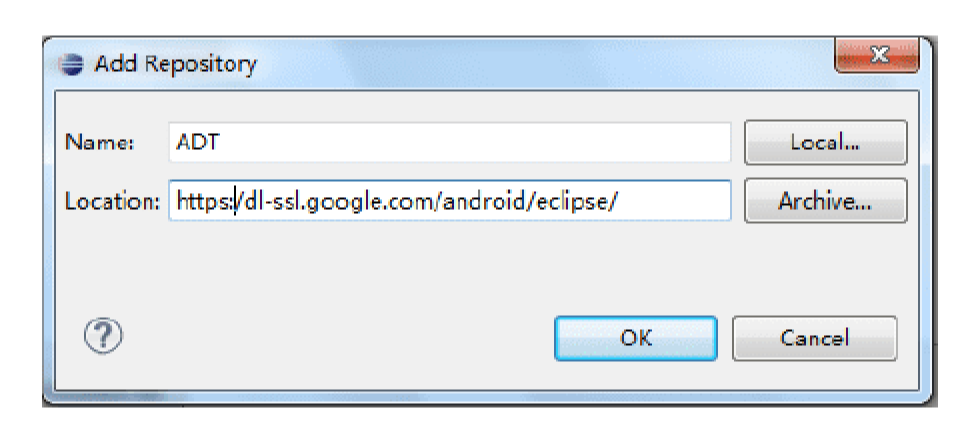


图 2-8　Add Repository 对话框

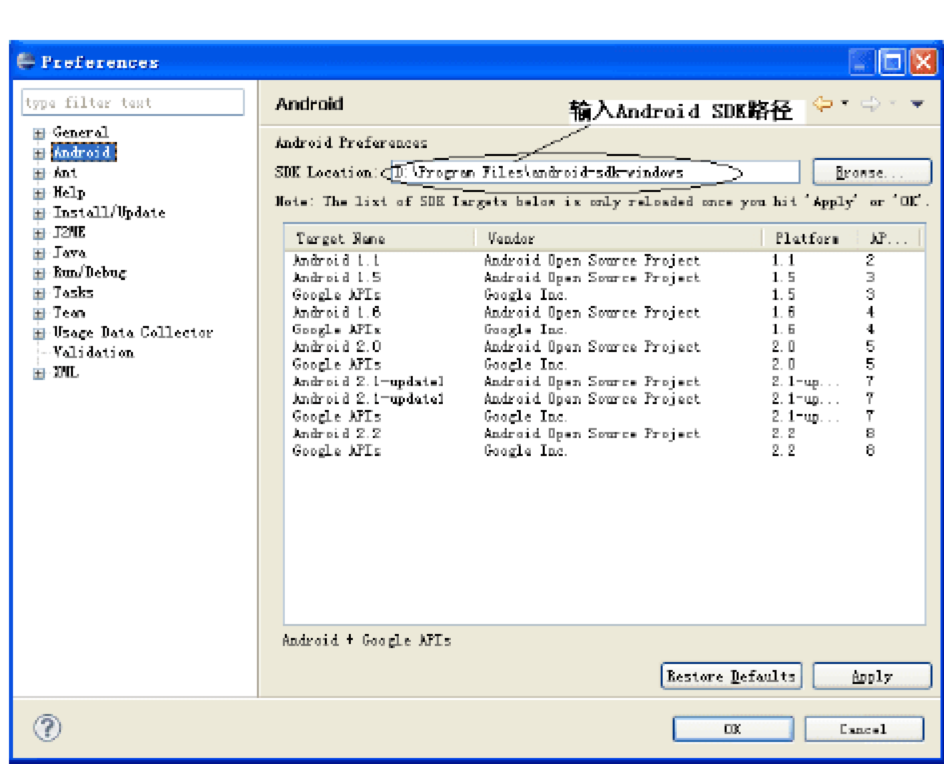


图 2-9　Preferences 窗口



说明

在上面介绍了 Android 应用开发平台的搭建，整个过程可以说比较繁琐，尤其在安装 ADT 时，经常会失败。现在 Google 公司对 Android 的开发工具进行了整合。在整合后的开发工具中，集成了以下内容：Eclipse、ADT 插件、Android Platform-tools、Android 平台、最新的模拟器镜像。开发人员可以从 http://developer.android.com/sdk/index.html#download 上下载最新版本的 Android SDK。下载时，需要同意用户协议，并且选择 32 位版本还是 64 位版本。在下载完毕后，只要解压该工具包即可。在解压后的文件夹中，运行 SDK Manager.exe 即可获取 Android 其他版本的 SDK。

除此之外，Google 在 2013 年还推出了全新的 Android 开发环境 Android Studio，该开发环境基于 IntelliJ IDEA。类似于 Eclipse ADT，Android Studio 提供了集成的 Android 开发工具用于开发和调试，在 IDEA 的基础上，Android Studio 提供以下功能：

（1）基于 Gradle 的构件支持。

（2）Android 专属的重构和快速修复。

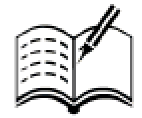


（3）提示工具以捕获性能、可用性、版本兼容性等问题。

（4）支持 ProGuard 和应用签名。

（5）基于模板的向导来生成常用的 Android 应用设计和组件。

（6）功能强大的布局编辑器，可以拖拉 UI 控件并进行效果预览。



Note

5. 创建虚拟设备

Android 为开发人员提供了可以在计算机上直接对应用程序进行测试的虚拟设备 AVD（Android Virtual Device），或称为模拟器。开发人员就可以直接在计算机上，而不用在 Android 智能手机上对程序进行调试。在 Eclipse 环境下创建 AVD 的步骤如下：

（1）启动 Eclipse，选择 Window/Android SDK and AVD Manager 命令。

（2）单击 New 按钮，弹出如图 2-10 所示的对话框。

（3）输入 AVD 名称、Android SDK 版本、SD 卡大小，单击 Create AVD 按钮，完成 AVD 的创建。创建成功的 AVD 将会显示在 Virtual Devices 列表中，如图 2-11 所示。

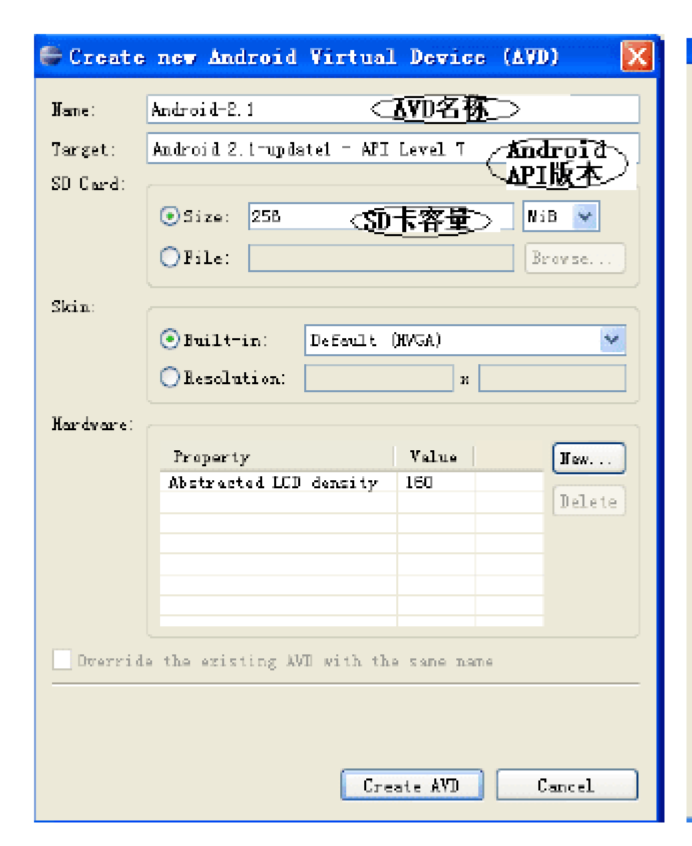


图 2-10　创建 AVD 对话框

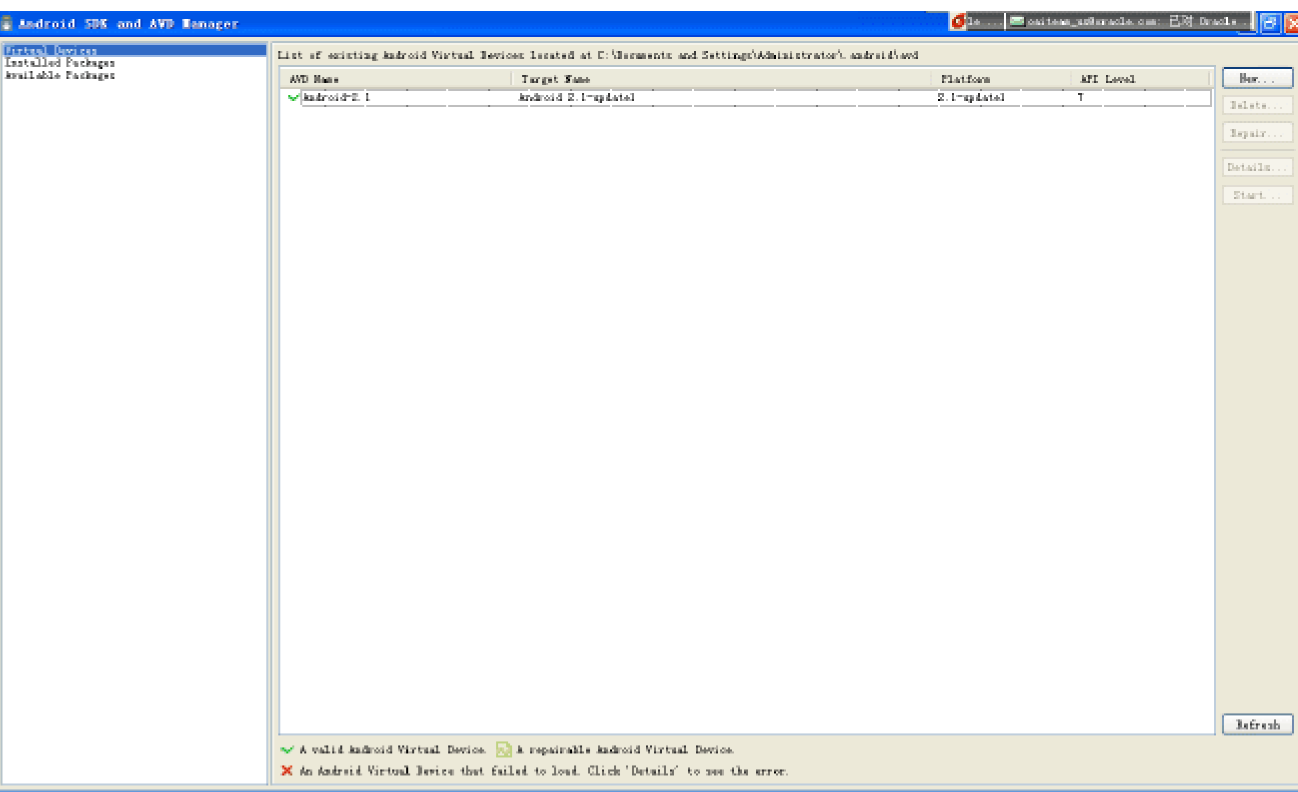


图 2-11　Virtual Devices 列表

创建的 AVD 文件默认存放在 C:\Documents and Settings\Administrator\.android\avd 目录下。在该文件夹下有所创建的 AVD 对应的文件夹及配置文件，在本书中为 Android-2.1.avd 文件夹及 Android-2.1.ini 文件。

如果需要把 AVD 的文件存放在其他位置，只需要把 Android-2.1.avd 复制到其他位置，例如 F 盘下，同时修改 Android-2.1.ini 文件为 target=android-7，path=F:\Android-2.1.avd 即可。

2.3　创建 Hello Android 项目

在 2.2 节，已经成功搭建了 Android 应用程序的开发平台，下面就开始 Android 应用程序的开发之旅。首先，创建一个 Android 项目——Hello Android。通过创建该项目，介

绍创建 Android 项目的过程。

创建 Hello Android 应用程序的步骤如下：

（1）启动 Eclipse，选择 File/New/Android Project 命令，将弹出创建新项目窗口，如图 2-12 所示。

（2）输入各项内容，单击 Finish 按钮，完成项目的创建。在图 2-12 中，如果输入内容不正确或者不符合要求，则会进行相应的提示。对于创建的 Hello Android 项目，不需要经过任何调试即可运行。

（3）运行该应用程序。在 Hello Android 项目上右击，依次选择 Run As/Android Application 命令，如图 2-13 所示。如果在此前没有创建 AVD，则系统会提示没有 AVD 可以运行，如图 2-14 所示。此时，需要单击 Yes 按钮创建 AVD。

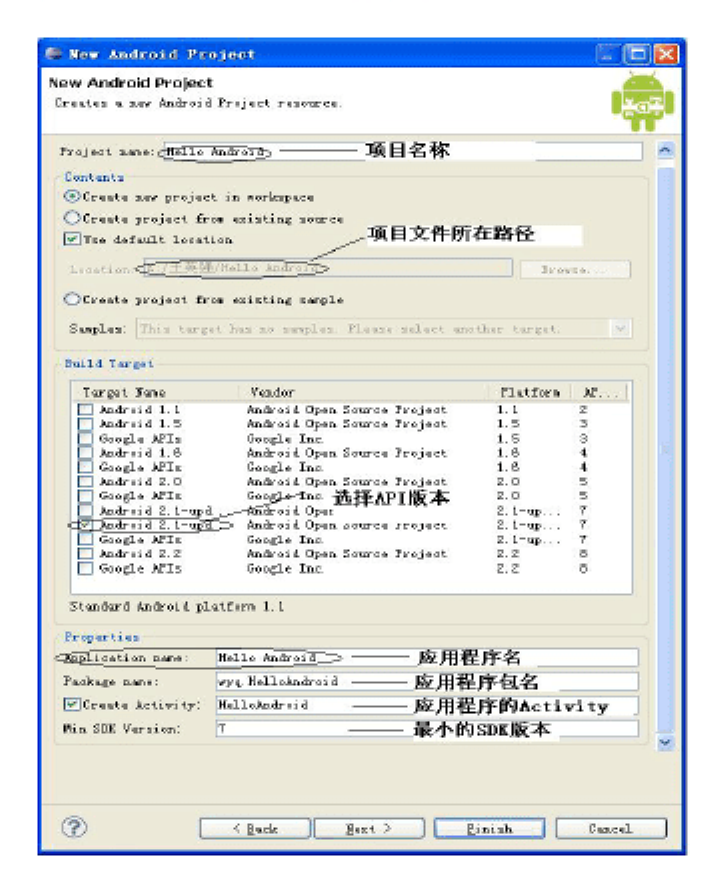


图 2-12　创建 Android 项目

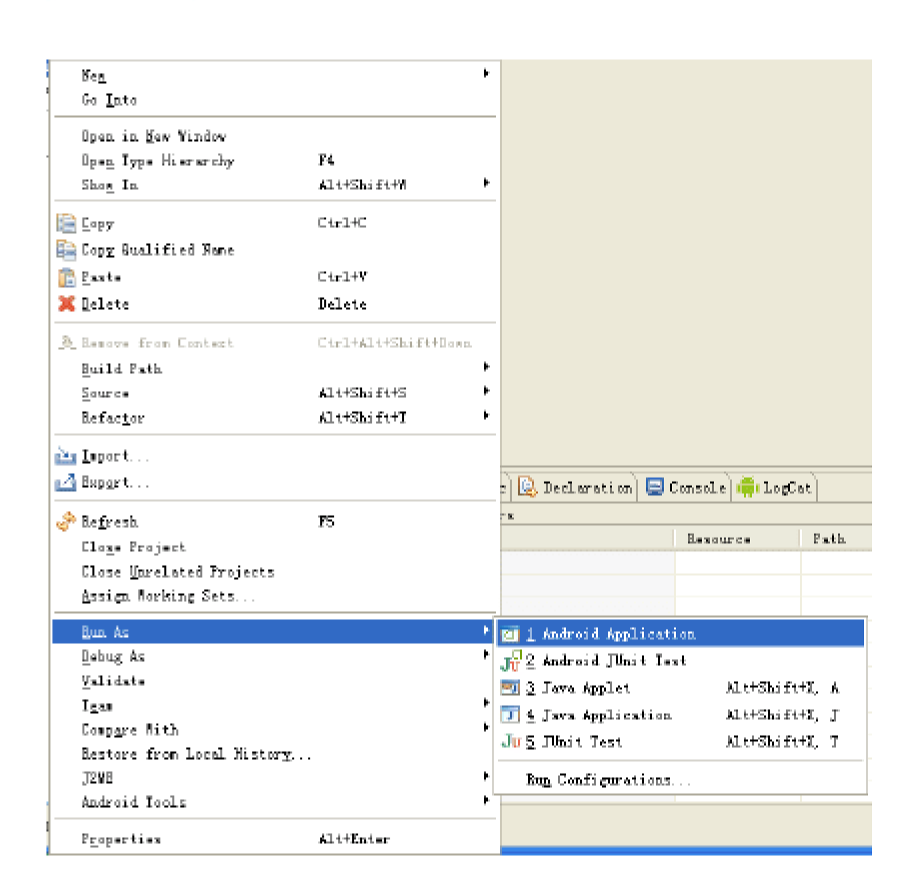


图 2-13　运行 Android 项目

（4）运行结果如图 2-15 所示。

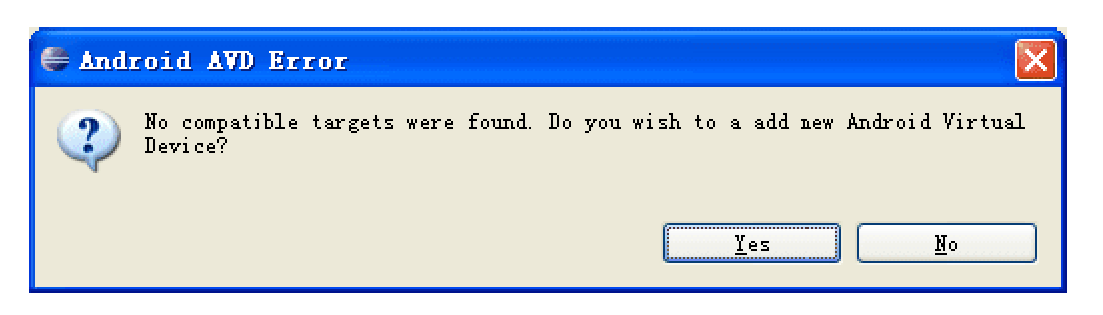


图 2-14　AVD 错误信息

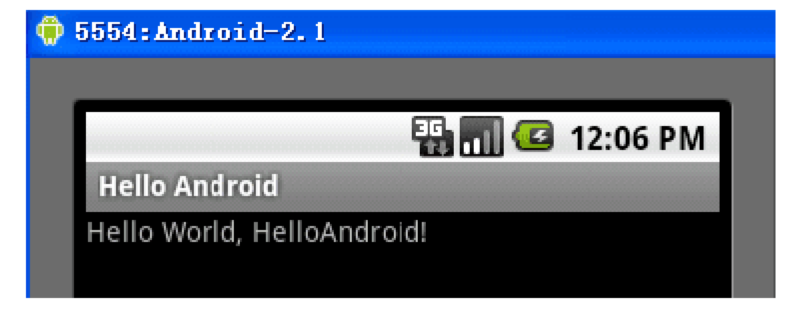
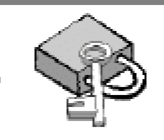


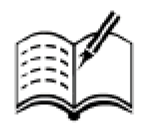
图 2-15　Hello Android 运行结果

2.4　Android 应用程序构成

本节主要介绍 Hello Android 应用程序的构成，让读者进一步了解 Android 应用程序文件的组成。展开 Package Explorer 窗口的 Hello Android 项目，对应的程序文件结构如图 2-16 所示。下面对主要文件夹及文件的作用进行介绍。

（1）src：用于存放应用程序的源代码。在图 2-16 中，wyq.HelloAndroid 为应用程序包，HelloAndroid.java 为应用程序的源代码，HelloAndroid 为该源代码文件中的类。





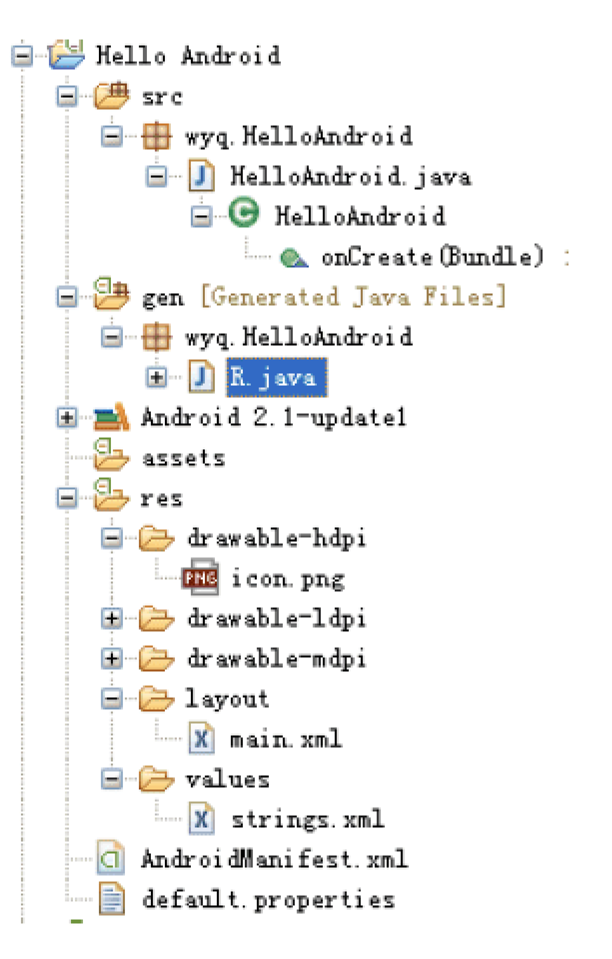


图 2-16　程序文件结构图

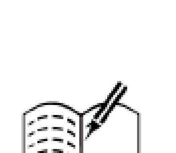
（2）gen：用于存放系统自动生成的类 R.java。在 R.java 文件中，为 res 文件夹下的每一个资源自动生成唯一的 ID。R.java 文件是 ADT 插件自动生成的，一般不需要进行修改。下面对 Hello Android 应用程序中的 R.java 文件进行解析。

```
1    package wyq.HelloAndroid;
2    public final class R {
3        public static final class attr {
4        }
5        public static final class drawable {
6        public static final int icon=0x7f020000;
7        }
8        public static final class layout {
9        public static final int main=0x7f030000;
10       }
11       public static final class string {
12       public static final int app_name=0x7f040001;
13       public static final int hello=0x7f040000;
14       }
15   }
```

说明：

- 第 1 行：说明 R.java 文件包含在 wyq.HelloAndroid 包中。
- 第 6 行：说明 icon.png 图片的 ID 号。icon.png 图片位于 res\drawable-hdpi 文件夹下。
- 第 9 行：说明 main 布局文件的 ID 号。main 布局文件为 res\layout 文件夹下的 main.xml 文件。
- 第 12~13 行：说明字符串 app_name 与 hello 的 ID 号，这两个字符串保存在 res\values 文件夹下的 string.xml 文件中。

（3）Android 2.1_update1：该文件夹下是一个 Android.jar 文件，该文件是为开发人员提供的一个类包，在应用程序开发过程中所引用的 Android 类都来自于此文件。



Note

（4）assets：该文件夹下可以存放应用程序所用到的任何文件，在该文件夹下放置的文件不会生成唯一的 ID 号。

（5）res：该文件夹下存放各种资源文件，对于每一个资源，都会在 R.java 文件中自动生成一个唯一的 ID 号。在 res 文件夹下有 drawable 系列文件夹、layout 文件夹和 values 文件夹。

① drawable 系列文件夹：存放应用程序使用到的图片，包括：drawable-hdpi（高分辨率）、drawable-ldpi（低分辨率）、drawable-mdpi（中等分辨率）3 个文件夹，用于存放不同分辨率的图片。因为 Android 手机的型号很多，屏幕的大小、分辨率都不同，如果只有一种分辨率的图片，可能导致在显示图片时不能正常显示。所以，在准备图片时，需要准备高、中、低 3 种分辨率版本的图片。

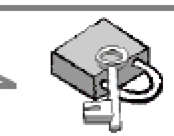
② layout 文件夹：存放每一个 Activity 的布局文件，每一个 Activity 都对应一个布局文件。布局文件是一个.xml 文件，用于控制 Activity 中每一个控件的位置、大小、字体、颜色等。在后面章节将详细介绍 Android 应用程序的布局方式及布局文件的使用。下面对 Hello Android 应用程序中 main.xml 布局文件进行解析。

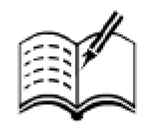
```
1    <?xml version="1.0" encoding="utf-8"?>
2    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3        android:orientation="vertical"
4        android:layout_width="fill_parent"
5        android:layout_height="fill_parent"
6    >
7    <TextView
8        android:layout_width="fill_parent"
9        android:layout_height="wrap_content"
10       android:text="@string/hello"
11   />
12   </LinearLayout>
```

说明：

- 第 1 行：说明.xml 文件的版本及编码方式。
- 第 2~6 行：说明该 Activity 的布局方式。第 2 行说明布局方式为线性布局；第 3 行说明布局的方向为垂直方向；第 4 行说明布局的宽度为整个父窗口；第 5 行说明布局的高度为整个父窗口。因为该布局为最上层的布局，则父窗口为手机的整个屏幕，所以该布局所对应的 Activity 会布满整个手机屏幕。
- 第 7~11 行：说明在该 Activity 中使用的控件。第 7 行说明该控件为文本控件；第 8 行说明文本控件的宽度为整个父窗口；第 9 行说明文本控件的高度为自适应文本的高度；第 10 行说明文本控件中显示的文字内容。@string/hello 引用的是 string.xml 文件中 hello 所对应的值。

③ values 文件夹：该文件夹下有 strings.xml 文件，存放应用程序中所使用的一些值，该文件同样是一个.xml 文件。在该文件中存放的都是一些键值对。下面对 Hello Android 应用程序的 string.xml 文件进行解析。






```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3  <string name="hello">Hello World, HelloAndroid!</string>
4  <string name="app_name">Hello Android</string>
5  </resources>
```

说明：

- 第 3 行：一个键值对，name 为 hello，对应的值为“Hello World, HelloAndroid!”，在 main 布局文件中，@string/hello 引用的就是该字符串。
- 第 4 行：一个键值对，name 为 app_name，对应的值为 Hello Android。

（6）AndroidManifest.xml 文件：该文件为整个应用程序的配置文件。下面对 AndroidManifest.xml 文件进行解析。

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="wyq.HelloAndroid"
4        android:versionCode="1"
5        android:versionName="1.0">
6  <application android:icon="@drawable/icon" android:label="@string/app_name">
7    <activity android:name=".HelloAndroid"
8        android:label="@string/app_name">
9        <intent-filter>
10               <action android:name="android.intent.action.MAIN" />
11               <category android:name="android.intent.category.LAUNCHER" />
12       </intent-filter>
13   </activity>
14   </application>
15   <uses-sdk android:minSdkVersion="7" />
16   </manifest>
```

说明：

- 第 2 行：说明 AndroidManifest.xml 文件的根标签是 manifest。
- 第 3 行：说明应用程序的包名。
- 第 6~14 行：为对应用程序的配置。第 6 行设置 android:icon 用来配置应用程序的图标，android:label 用来配置应用程序的标签；第 7~13 行为 Activity 的配置，第 7 行 android:name 用来配置该 Activity 对应的类名；第 8 行 android:label 用来配置该 Activity 的标签；第 9~12 行用来配置该 Activity 为整个应用程序的起始 Activity，即首界面。
- 第 15 行：配置应用程序最低的 SDK 版本。

2.5 习　　题

1．根据 2.2 节内容，在个人计算机上搭建 Android 开发平台。

2．创建一个 MyFirstAndroid 项目。

3．简述 MyFirstAndroid 项目中各组成文件的作用。

第 3 章 Activity 组件

【本章内容】

- Activity 简介
- 调用其他的 Activity
- 不同 Activity 之间的数据传送
- 返回数据到前一个 Activity
- Activity 的生命周期与管理

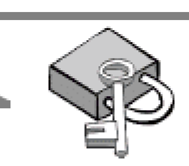
本章主要介绍 Android 应用程序开发中最常用的组件——Activity。可以认为一个界面或者一个窗口就是一个 Activity。在 Activity 中可以通过摆放各种控件来设计应用程序的用户界面。当一个应用程序有多个用户界面时，Activity 之间的调用与数据之间的传送则是开发人员必须掌握的内容，本章将介绍如何调用其他的 Activity，以及 Activity 之间的数据传送。同时，介绍 Activity 的运行机制及生命周期，这将有助于读者对 Activity 更好地理解与使用。

3.1 Activity 简介

对于具有用户界面的应用程序来说，至少有一个 Activity。在理解什么是 Activity 时，最简单的方法就是将应用程序的一个界面与某个 Activity 联系起来，因为 Activity 与用户界面之间多为一对一的关系，每个 Activity 显示一个用户界面并响应一些系统和用户发起的事件。用户可以通过将 Activity 类进行扩展，即用户的 Activity 类派生于 Android SDK 提供的 Activity 类，来完成用户界面类的设计与实现。

第 2 章的 Hello Android 项目中，实现了一个简单的 Activity 设计。下面对该 Activity 的源代码进行解析。

```
public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```



说明：

- 第 1 行：说明应用程序的界面对应的类 HelloAndroid 派生于 Activity，即对 Activity 进行扩展。
- 第 5 行：调用父类的 onCreate()构造函数，savedInstanceState 是保存当前 Activity 的状态信息。
- 第 6 行：设置用户界面，该用户界面采用的布局文件为 main.xml（main.xml 文件源代码的解析详见 2.4 节）。R.layout.main 是 Android 调用资源的方法，调用的是 res\layout\main.xml 资源。

Activity 类是 Android 运行时 Android.jar 包 android.app 的一部分，在 Android 中表示可见度非常高的应用程序组件，通过与 View 类结合使用，来显示用户界面。

3.2　调用其他的 Activity

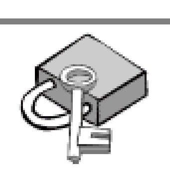
在一个应用程序中，可能存在多个操作界面，则界面之间难免存在调用关系。下面通过一个实例来演示在一个 Activity 中如何调用另外一个 Activity。

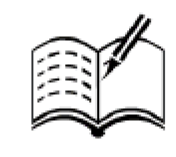
创建 EX03_1 项目的步骤如下：

（1）创建 EX03_1 项目，操作与创建 Hello Android 相同。

（2）修改主 Activity 的布局文件 main.xml，增加一个命令按钮。源代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="第一个 Activity"
 />
 <EditText
     android:id="@+id/name"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
 />
 <Button
     android:id="@+id/bt1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="调用第二个 Activity"
 />
 </LinearLayout>
```





Note

说明：

- 第 12~16 行：声明一个 EditText 控件。第 13 行 android:id 为设置文本框的 ID；第 14～15 行设置 Edit Text 控件。
- 第 17~22 行：定义一个 Button 命令按钮控件。第 18 行 android:id 为设置命令按钮的 ID；第 21 行 android:text 为设置命令按钮的文本。

（3）创建第二个 Activity 的布局文件 second.xml。方法如下：

① 在 res\layout 文件夹上右击，选择 New/File 命令。

② 在 filename 文本框中输入 second.xml。

③ 打开 second.xml 文件，编辑代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
<TextView
    android:id="@+id/tv"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="这是第二个 Activity"
/>
</LinearLayout>
```

（4）增加第二个 Activity 的类文件。方法如下：

① 在 src\wyq.EX03_1 文件夹上右击，选择 New/Class 命令。

② 在 Name 文本框中输入第二个 Activity 对应的类名 SecondActivity。

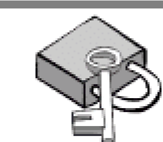
③ 打开文件，编辑代码如下：

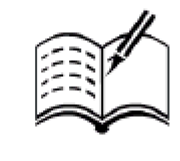
```
1   package wyq.EX03_1;
2   import android.app.Activity;
3   import android.os.Bundle;
4    public class SecondActivity extends Activity {
5           /** Called when the activity is first created. */
6           @Override
7           public void onCreate(Bundle savedInstanceState) {
8           super.onCreate(savedInstanceState);
9           setContentView(R.layout.second);
10          }
11  }
```

说明：

第 9 行：为设置第二个 Activity 的布局文件。

（5）修改 AndroidManifest.xml 文件，为第二个 Activity 进行配置。在该文件的<application>节点中增加如下代码：






```
<activity android:name=".SecondActivity"
        android:label="@string/app_name">
</activity>
```

android:name 为第二个 Activity 对应的类名，注意要区分大小写。

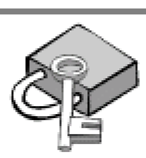
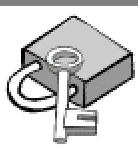
（6）修改 FirstActivity 的类文件，为该 Activity 的命令按钮增加单击监听事件。编辑代码如下：

```
1   package wyq.EX03_1;
2   import android.app.Activity;
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.view.View;
6   import android.widget.Button;
7   public class FirstActivity extends Activity {
8       /** Called when the activity is first created. */
9       private Button bt;
10      @Override
11      public void onCreate(Bundle savedInstanceState) {
12          super.onCreate(savedInstanceState);
13          setContentView(R.layout.main);
14          bt=(Button)findViewById(R.id.bt1);
15          bt.setOnClickListener(new Button.OnClickListener()
16          {
17              public void onClick(View v)
18              {
19                  Intent intent=new Intent();
20                  intent.setClass(FirstActivity.this, SecondActivity.class);
21                  startActivity(intent);
22              }
23          }
24          );
25      }
26  }
```

说明：

- ❑ 第 9 行：声明一个 Button 类变量。
- ❑ 第 14 行：使用 findViewById()获取 Button 对象。
- ❑ 第 15~24 行：为 Button 添加单击监听事件。第 19 行定义 Intent 对象；第 20 行调用 Intent 类的 setClass()函数，指定要启动的 class（setClass()的第二个参数）；第 21 行调用一个新的 Activity。

项目 EX03_1 的运行结果如图 3-1 所示，单击命令按钮，显示第二个 Activity，如图 3-2 所示。

Note





图 3-1　EX03_1 运行结果



图 3-2　第二个 Activity

3.3　不同 Activity 之间的数据传送

在 3.2 节的实例中，介绍了在一个 Activity 中如何调用另外一个 Activity。在实际的开发工程中，有时需要在调用另外一个 Activity 的同时，传递一些数据。对于这种情况，就需要利用 Android.os.Bundle 对象封装数据，通过 Bundle 对象在不同的 Intent 之间传递数据。

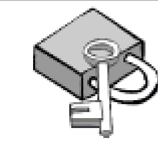
在本节实例中，将对 3.2 节的实例进行扩展修改：在第一个 Activity 的文本框中输入内容，然后把文本框中的内容传送到第二个 Activity，并且进行显示。

创建 EX03_2 项目，步骤如下：

（1）按创建 EX03_1 的方法的前 5 步进行操作。

（2）修改 FirstActivity 的类文件，为该 Activity 的命令按钮增加单击监听事件。主要代码如下：

```
package wyq.EX03_2;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private Button bt;
    private EditText name;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bt=(Button)findViewById(R.id.bt1);
```



```
17                name=(EditText)findViewById(R.id.name);
18                bt.setOnClickListener(new Button.OnClickListener()
19                {
20                    public void onClick(View v)
21                    {
22                        String myName=name.getText().toString();
23                        Intent intent=new Intent();
24                        intent.setClass(FirstActivity.this, SecondActivity.class);
25                        Bundle bundle=new Bundle();
26                        bundle.putString("name", myName);
27                        intent.putExtras(bundle);
28                        startActivity(intent);
29                        }
30                }
31                );
32            }
33    }
```

说明：

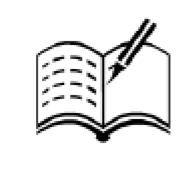
- ❑ 第 11 行：定义一个文本框变量。
- ❑ 第 16 行：使用 findViewById()获取 EditText 对象。
- ❑ 第 17 行：获取文本框的输入内容。
- ❑ 第 25~26 行：定义一个 Bundle 对象，并将要传递的数据传入。bundle.putString () 函数传递的是一个键值对，name 为键名，myName 为键值，即要传递的数据。
- ❑ 第 27 行：将 Bundle 对象传递给 Intent。

（3）修改 SecondActivity.java 文件，编写代码如下：

```
1    package wyq.EX03_2;
2    import android.app.Activity;
3    import android.os.Bundle;
4    import android.widget.TextView;
5    public class SecondActivity extends Activity {
6        /** Called when the activity is first created. */
7        private TextView tv;
8        @Override
9        public void onCreate(Bundle savedInstanceState) {
10           super.onCreate(savedInstanceState);
11           setContentView(R.layout.second);
12           Bundle bundle=this.getIntent().getExtras();
13           String myName=bundle.getString("name");
14           tv=(TextView)findViewById(R.id.tv);
15           tv.setText("欢迎"+myName+"来到 Android 世界");
16       }
17   }
```

说明：

- ❑ 第 12 行：取得 Intent 中的 Bundle 对象。


Note

- 第 13 行：取得 Bundle 对象中的数据。
- 第 14 行：使用 findViewById()获取 TextView 对象。
- 第 15 行：设置文本标签的内容。

实例 EX03_2 运行结果如图 3-3 所示，单击命令按钮，显示第二个 Activity，如图 3-4 所示。



图 3-3　EX03_2 运行结果



图 3-4　第二个 Activity

3.4　返回数据到前一个 Activity

在 3.3 节的实例中，我们将数据从第一个 Activity 传递到第二个 Activity，完成了 Activity 之间数据的传递。如果在程序的运行过程中，又要返回到上一个页面，会发生什么情况呢？

在访问 Internet 时，可以通过后退键来返回到上一个访问页面，而在 Android 应用程序中，则可以通过手机的返回键来完成。但是，如果在应用程序的界面上增加一个“返回上一页”按钮，通过该按钮返回上一页，且返回后要能够保留之前输入的相关信息，那么就必须使用 startActivityForResult()来调用另一个 Activity。使用该方法，第一个 Activity 便会有一个等待第二个 Activity 的返回，就可以达到我们想要的结果。在本节中创建 EX03_3 项目，步骤如下：

（1）按创建 EX03_1 的方法的前 5 步进行操作。

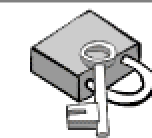
（2）修改 second.xml 文件，在第二个 Activity 上增加一个 Button。代码如下：

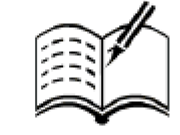
```
<Button
android:id="@+id/returnBack"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="返回上一页"
/>
```

（3）修改 FirstActivity.java 文件。

① 修改 EX03_2 项目中 FirstActivity.java 的源代码中命令按钮的单击监听事件，将 startActivity(intent)修改为 startActivityForResult(intent,0)。startActivityForResult(Intent intent, Int requestCode)函数的第一个参数为 Intent 对象，第二个参数 requestCode 是一个大于等于 0 的整数，用于在 onActivityResult()函数中区别哪个子模块回传的数据。

② 重写 onActivityResult 函数，代码如下：






```
1   protected void onActivityResult(int requestCode, int resultCode, Intent data) {
2       // TODO Auto-generated method stub
3       switch(resultCode)
4       {
5           case RESULT_OK:
6           Bundle bundle=data.getExtras();
7           String myName=bundle.getString("name");
8           name.setText(myName);
9           break;
10          default: break;
11      }
12  }
```

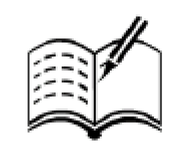
说明：

- 第 5 行：从 SecondActivity 中返回的 resultCode 是 RESULT_OK。
- 第 6 行：取得来自 SecondActivity 的数据，并显示在 FirstActivity 的文本框中。

（4）修改 SecondActivity.java 文件，编写代码如下：

```
package wyq.EX03_3;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class SecondActivity extends Activity {
    /** Called when the activity is first created. */
    private TextView tv;
    private Button returnBack;
    Intent intent;
    Bundle bundle;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
        intent=this.getIntent();
        bundle=intent.getExtras();
        String myName=bundle.getString("name");
        tv=(TextView)findViewById(R.id.tv);
        tv.setText("欢迎"+myName+"来到 Android 世界");
        returnBack=(Button)findViewById(R.id.returnBack);
        returnBack.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                SecondActivity.this.setResult(RESULT_OK,intent);
                SecondActivity.this.finish();
            }
        });
```

```
32        }
33    }
```


Note

说明：

- ❑ 第 18 行：取得 Intent 中的 Bundle 对象。
- ❑ 第 23 行：使用 findViewById()取得 Button 对象。
- ❑ 第 24 行：为 Button 增加单击监听事件。
- ❑ 第 28 行：返回 result 到上一个 Activity。
- ❑ 第 29 行：结束本 Activity。

实例 EX03_3 运行结果如图 3-5 所示，单击命令按钮，结果如图 3-6 所示。单击图 3-6 中的命令按钮，返回第一个 Activity，结果如图 3-7 所示。




图 3-5　EX03_3 运行结果

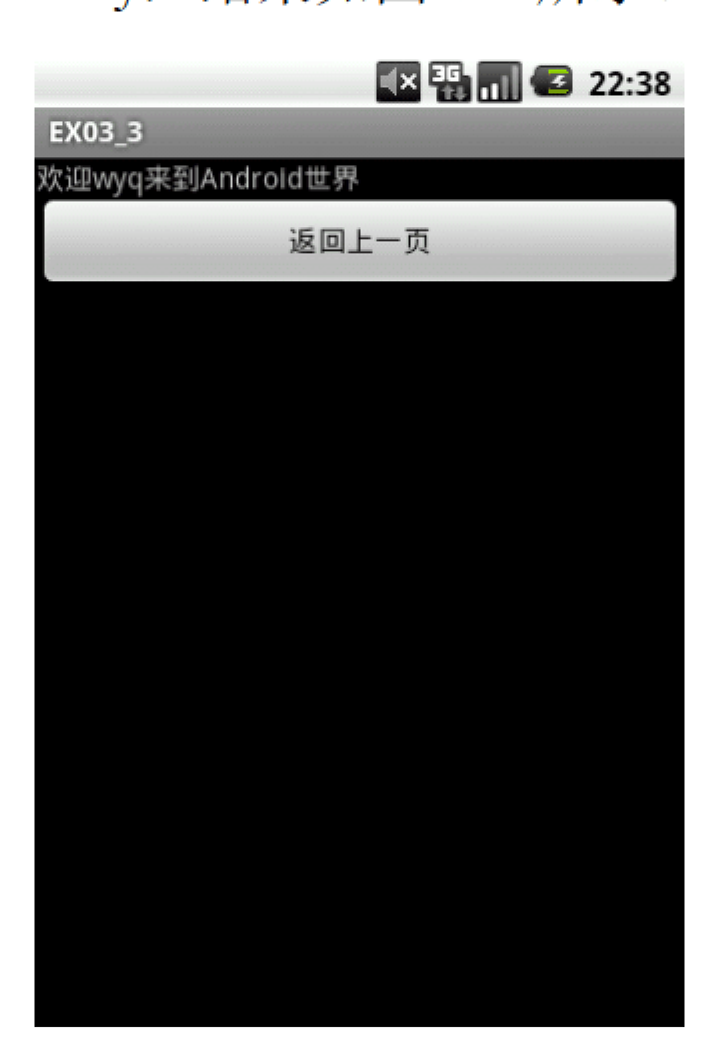


图 3-6　第二个 Activity

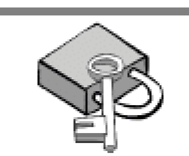


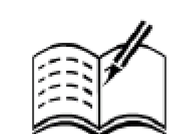
图 3-7　返回结果

3.5　Activity 的生命周期与管理

在一个 Android 程序中，至少要有一个 Activity。Activity 是一个对象，也可以想象成有生命形式存在的一种方式，有“生老病死”的过程。Activity 的各种状态之间的切换通过 7 个生命周期方法来实现：onCreate()、onStart()、onResume()、onPause()、onStop()、onDestroy()、onRestart()，每个方法的作用如下所述。

- ❑ onCreate()：当一个 Activity 第一次被创建时就会调用该方法，这时可以初始化数据，例如为 ListView 绑定数据。
- ❑ onStart()：当一个 Activity 可以被用户看到时就会调用该方法。
- ❑ onResume()：在 Android 应用程序中，所有的 Activity 都存放在一个 Activity 堆栈里面。所谓的栈就是遵循 LIFO（last in first out）规律的存储空间，对于这段 Activity 的存储空间只有两种操作——入栈与出栈，所以对于放在最顶上的 Activity 总是最先被看到。onResume()就是当这个 Activity 被置于栈顶时调用的方法。





- ❑ onPause():当启动另一个 Activity 时会调用此方法,新的 Activity 会把旧的 Activity 遮住。当旧的 Activity 被局部遮住,用鼠标单击不到的情况下就会调用 onPause();如果时间久了,原来被遮住的 Activity 都会消失,可以理解为线程挂起的状态。
- ❑ onStop():该方法与 onPause()方法的区别就在于当一个 Activity 被完全遮住时就会调用该方法。
- ❑ onDestroy():该方法用来销毁 Activity,同样的,finish()方法也会调用 onDestroy()方法销毁 Activity。
- ❑ onRestart():当再次启动 Activity 时就会调用该方法。

Android 使用堆栈对 Activity 进行管理,就是说某一个时刻只有一个 Activity 处在栈顶。当有一个新的 Activity1 被创建出来时,则新的 Activity1 将成为正在运行中的 Activity,而前一个 Activity2 保留在栈中。当用户按下后退按键,屏幕当前的 Activity1 将从栈中弹出,而 Activity2 恢复成运行中状态。

Activity 各生命周期方法之间的调用关系如图 3-8 所示,调用的时间点如图 3-9 所示。

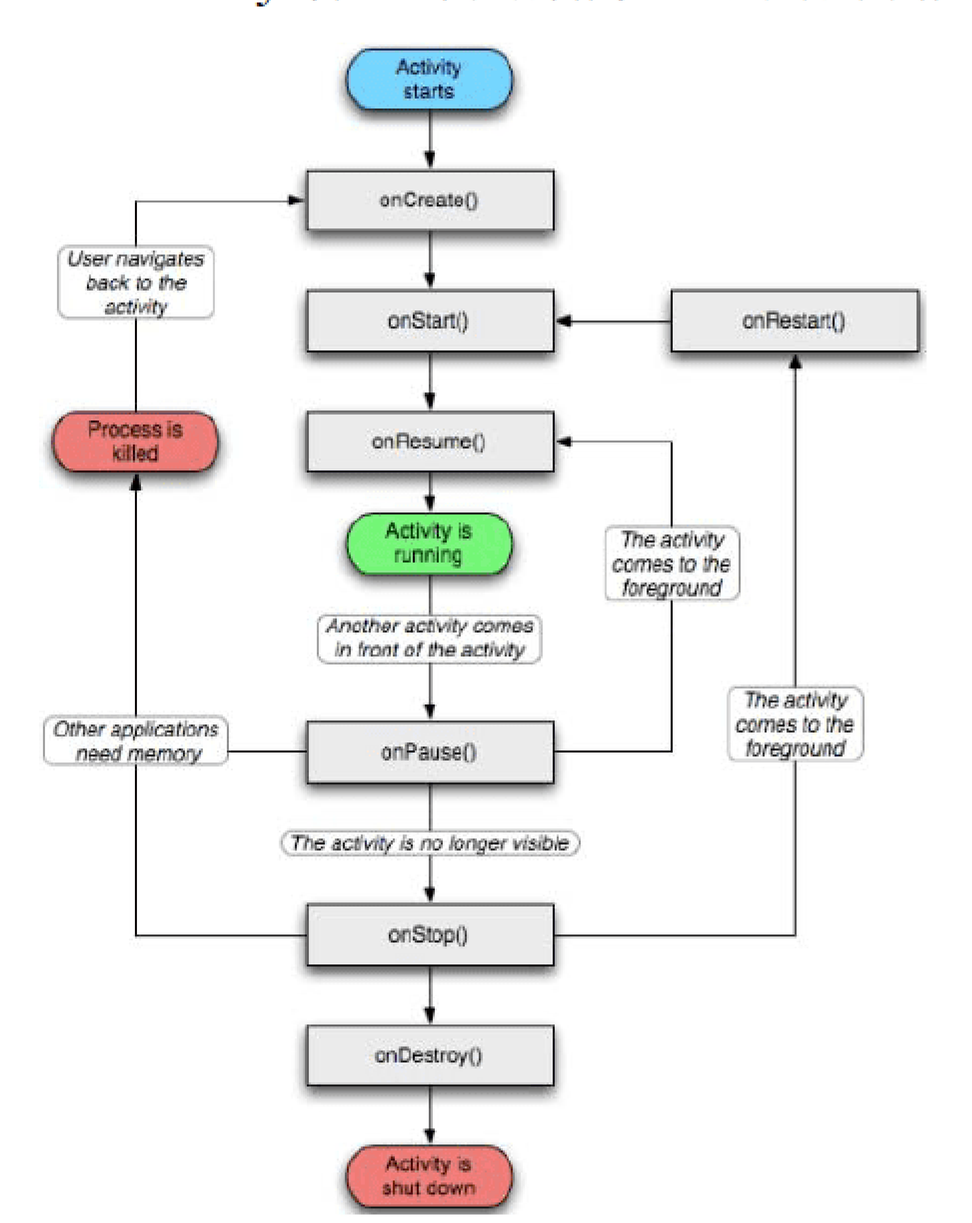


图 3-8　Activity 生命周期方法图

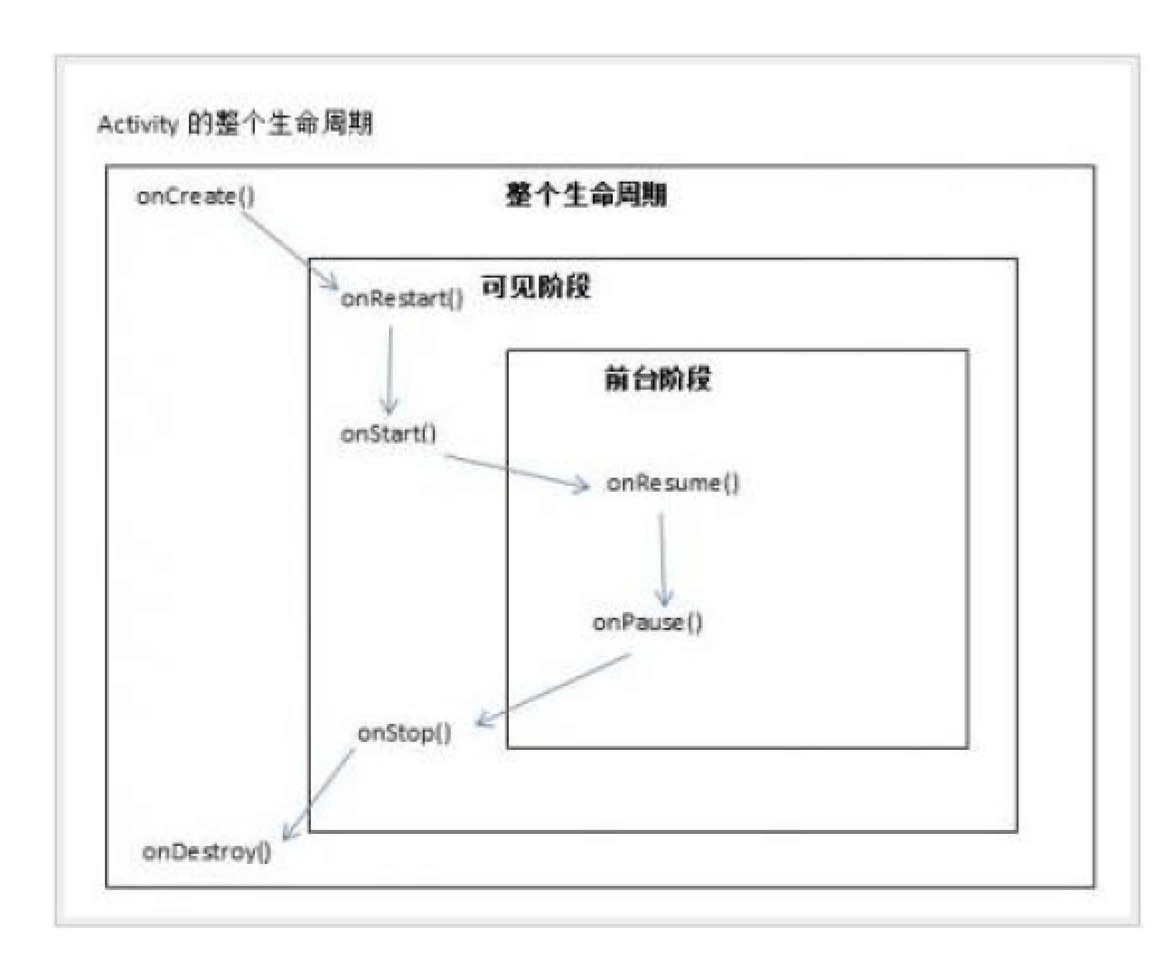


图 3-9　Activity 生命周期各个方法调用的时间点

下面通过 EX03_4 项目演示 Activity 的生命周期方法的调用过程。在本项目中会用到两个 Activity,通过第一个 Activity 调用第二个 Activity,然后再返回到第一个 Activity,观察生命周期方法的调用过程。

创建 EX03_4 项目的步骤如下:

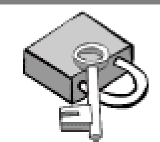
(1)创建 EX03_4 项目,步骤与 EX03_3 项目相同。

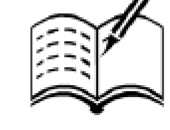
(2)修改 FirstActivity.java 代码文件,重写方法 onDestroy()、onPause()、onRestart()、onResume()、onStart()、onStop(),编写代码如下:

```
package wyq.EX03_4;
import android.app.Activity;
```


Note

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class FirstActivity extends Activity {
/** Called when the activity is first created. */
 private Button bt;
 private EditText name;
 @Override
 public void onCreate(Bundle savedInstanceState) {
     System.out.println("FirstActivity-->>onCreate");
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     bt=(Button)findViewById(R.id.bt1);
     name=(EditText)findViewById(R.id.name);
     bt.setOnClickListener(new Button.OnClickListener()
     {
         public void onClick(View v)
         {
             String myName=name.getText().toString();
             Intent intent=new Intent();
             intent.setClass(FirstActivity.this, SecondActivity.class);
             Bundle bundle=new Bundle();
             bundle.putString("name", myName);
             intent.putExtras(bundle);
             startActivityForResult(intent,0);
         }
     }
     );
 }
 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
     // TODO Auto-generated method stub
     switch(resultCode)
     {
     case RESULT_OK:
         Bundle bundle=data.getExtras();
         String myName=bundle.getString("name");
         name.setText(myName);
         break;
     default: break;
     }
 }
 @Override
 protected void onDestroy() {
     // TODO Auto-generated method stub
     System.out.println("FirstActivity-->>onDestroy");
     super.onDestroy();
```






```
52        }
53        @Override
54        protected void onPause() {
55            // TODO Auto-generated method stub
56            System.out.println("FirstActivity-->>onPause");
57            super.onPause();
58        }
59        @Override
60        protected void onRestart() {
61            // TODO Auto-generated method stub
62            System.out.println("FirstActivity-->>onRestart");
63            super.onRestart();
64        }
65        @Override
66        protected void onResume() {
67            // TODO Auto-generated method stub
68            System.out.println("FirstActivity-->>onResume");
69            super.onResume();
70        }
71        @Override
72        protected void onStart() {
73            // TODO Auto-generated method stub
74            System.out.println("FirstActivity-->>onStart");
75            super.onStart();
76        }
77        @Override
78        protected void onStop() {
79            // TODO Auto-generated method stub
80            System.out.println("FirstActivity-->>onStop");
81            super.onStop();
82        }
83    }
```

说明：

- ❑ 第 14 行：System.out.println("FirstActivity-->>onCreate");表示输出一条信息 FirstActivity-->>onCreate。当调用 OnCreate()函数时，就会输出该信息，用来跟踪 OnCreate()函数的调用情况。
- ❑ 第 48~52 行：重写 onDestroy()函数。
- ❑ 第 54~58 行：重写 onPause()函数。
- ❑ 第 60~64 行：重写 onRestart()函数。
- ❑ 第 66~70 行：重写 onResume()函数。
- ❑ 第 72~76 行：重写 onStart()函数。
- ❑ 第 78~82 行：重写 onStop()函数。

（3）修改 SecondActivity.java 文件，重写以下函数：onDestroy()、onPause()、onRestart()、onResume()、onStart()、onStop()，方法与 FirstActivity.java 相同。以 onCreate()函数为例，输出信息为 System.out.println("SecondActivity-->>onCreate")。

（4）使用 LogCat 查看程序的输出信息。

① 程序开始运行，显示第一个 Activity，输出信息为：

FirstActivity-->>onCreate

FirstActivity-->>onStart

FirstActivity-->>onResume


说明

在显示第一个 Activity 时，会调用第一个 Activity 的 OnCreate()、OnStart()、OnResume()函数，同时把第一个 Activity 放到 Activity 堆栈顶部。

② 单击命令按钮，调用第二个 Activity，输出信息为：

FirstActivity-->>onPause

SecondActivity-->>onCreate

SecondActivity-->>onStart

SecondActivity-->>onResume

FirstActivity-->>onStop


说明

调用第二个 Activity 时，第一个 Activity 会暂停，调用 OnPause()函数，然后显示第二个 Activity，调用第二个 Activity 的 OnCreate()、OnStart()、OnResume()函数。执行 OnResume()函数时，把第二个 Activity 放到 Activity 栈顶，第一个 Activity 会进行压入操作。当第二个

③ 单击命令按钮，返回第一个 Activity，输出信息为：

SecondActivity-->>onPause

FirstActivity-->>onRestart

FirstActivity-->>onStart

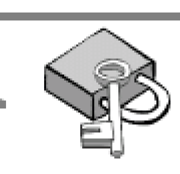
FirstActivity-->>onResume

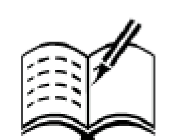
SecondActivity-->>onStop

SecondActivity-->>onDestroy


说明

当返回第一个 Activity 时，第二个 Activity 首先暂停，调用第二个 Activity 的 onPause()函数，然后第一个 Activity 会被重新启动并显示，所以依次调用 OnRestart()、OnStart()、OnResume()函数，第二个 Activity 则从 Activity 堆栈中弹出并销毁，所以依次调用 OnStop()、





3.6 习　　题

1．简述 Activity 组件。

2．新建一个 Android 项目，在该项目中实现以下功能：实现摄氏温度与华氏温度的转换。在 Activity1 中，输入摄氏温度，将计算结果传递至 Activity2 进行显示，如图 3-10 和图 3-11 所示。摄氏温度与华氏温度的转换公式为：

华氏温度 ＝32＋ 摄氏温度 ×1.8

摄氏温度 ＝(华氏温度−32)÷1.8



图 3-10　Activity1 界面

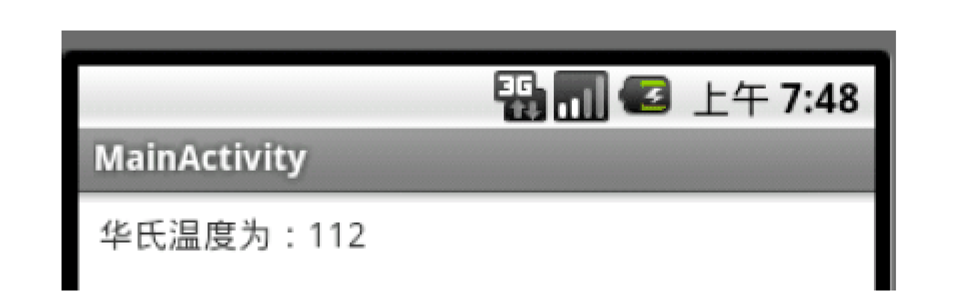


图 3-11　Activity2 界面

3．新建一个 Android 项目，在该项目中实现以下功能：在 Activity1 中，在文本控件中输入姓名与身高，使用单选按钮选择性别，将结果传递到 Activity2；在 Activity2 中，显示 Activity1 传输的数据，并且单击返回命令按钮，将数据返回到 Activity1，如图 3-12 和图 3-13 所示。



图 3-12　Activity1 界面



图 3-13　Activity2 界面

4．简述 Activity 的生命周期及其周期函数的作用。

第 4 章 Android 布局管理

【本章内容】

- View 布局概述
- 线性布局
- 表格布局
- 相对布局
- 帧布局
- 绝对布局
- 布局的嵌套

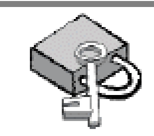
对于一个软件，漂亮的用户界面（UI）总能给使用者留下深刻的印象。对于 Android 手机应用软件而言，如何从众多的软件中脱颖而出，用户界面的设计是一个不可忽视的因素。在 Android 中，View 有五大布局方式：分别是线性布局（LinearLayout）、表格布局（TableLayout）、相对布局（RelativeLayout）、帧布局（FrameLayout）、绝对布局（AbsoluteLayout），布局方式使用 XML 语言进行描述。本章将对 Android 的五大布局方式进行介绍。

4.1 View 布局概述

在介绍 Android 的视图管理之前，首先需要了解一下 View 类。View 类是所有可视化控件的基类，主要提供控件绘制和事件处理的方法。前面的实例中所用到的 TextView、EditText、Button 均继承自 View 类。

View 类关系图显示了 View 类及其很多派生类的关系(没有包含 View 的全部派生类)，如图 4-1 所示。从图中可以看出 ViewGroup 类是一个与布局相关的、View 类的子类。

结合使用 View 基类方法和子类方法，可以设置布局、填充、焦点、高度、宽度、颜色等属性。关于 View 及其子类的相关属性，既可以在布局 XML 文件中使用“Android:名称空间”来设置，也可以通过成员方法在代码中进行设置。View 类常用的属性及其对应方法如表 4-1 所示。



Note

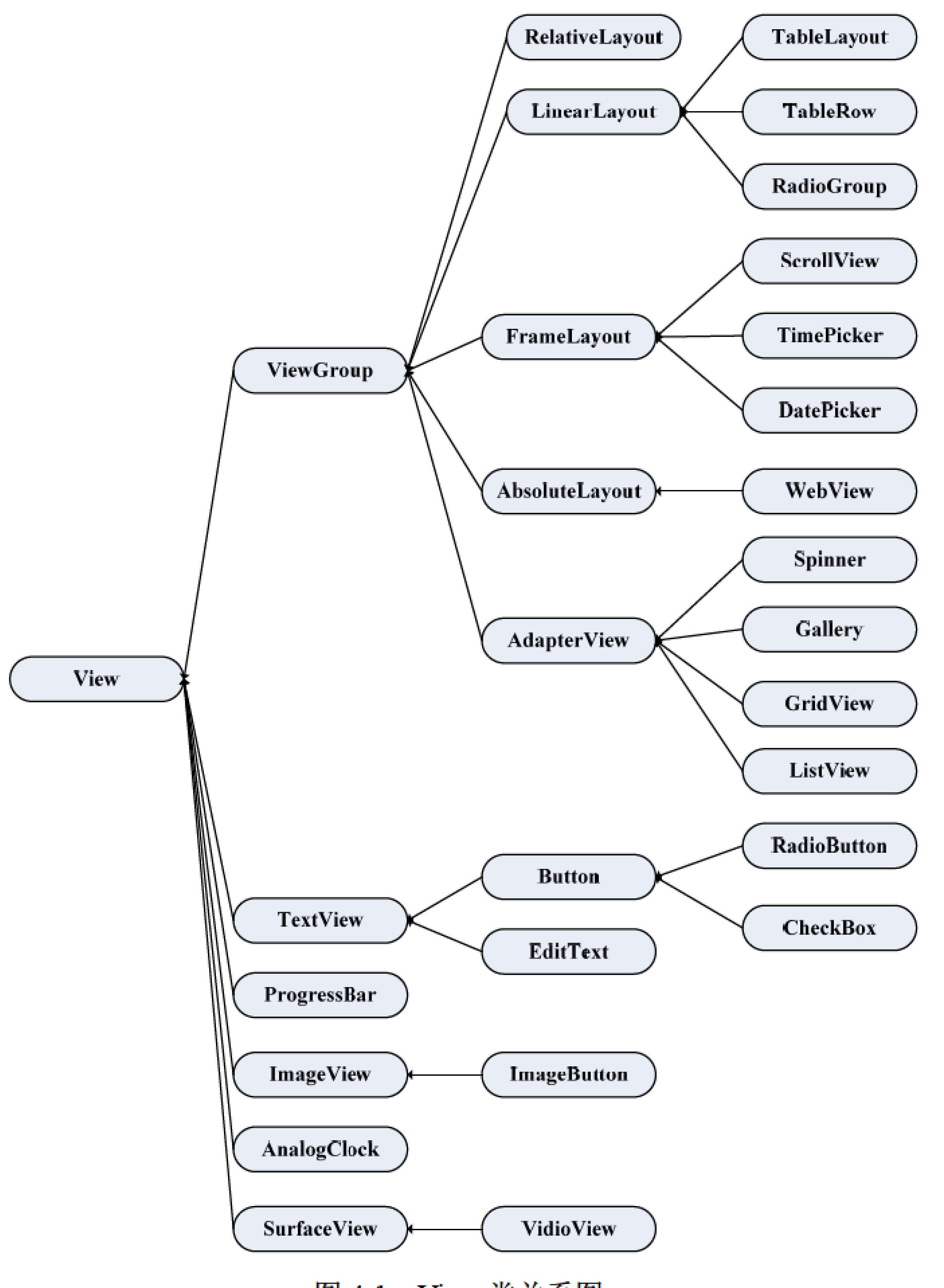
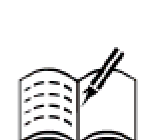


图 4-1　View 类关系图

表 4-1　View 类常用属性及对应方法

属性名称	对应方法	描　述
android:background	setBackgroundResource(int)	设置背景色/背景图片。可以通过以下两种方法设置背景为透明：@android:color/transparent 和 @null。注意 TextView 默认是透明的，不用写此属性，但是 Button、ImageButton 和 ImageView 想透明就要设置该属性了
android:focusable	setFocusable(boolean)	设置是否获得焦点。若有 requestFocus()被调用时，后者优先处理。注意在表单中想设置某一个控件，如 EditText 获取焦点，仅设置该属性是不行的，需要将 EditText 前面的 focusable 都设置为 false 才行。在 Touch 模式下获取焦点需要设置 focusableInTouchMode 为 true


Note

续表

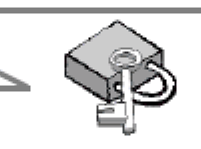
属性名称	对应方法	描　述
android:id	setId(int)	给当前 View 设置一个在当前 layout.xml 中的唯一编号，可以通过调用 View.findViewById()或 Activity.findViewById()，根据该编号查找到对应的 View。不同的 layout.xml 之间定义相同的 id 不会冲突。格式如@+id/btnName
android:minHeight		设置视图最小高度
android:minWidth		设置视图最小宽度
android:padding	setPadding(int,int,int,int)	设置上、下、左、右的边距，以像素为单位填充空白
android:paddingBottom	setPadding(int,int,int,int)	设置底部的边距，以像素为单位填充空白
android:paddingLeft	setPadding(int,int,int,int)	设置左边的边距，以像素为单位填充空白
android:paddingRight	setPadding(int,int,int,int)	设置右边的边距，以像素为单位填充空白
android:paddingTop	setPadding(int,int,int,int)	设置上方的边距，以像素为单位填充空白
android:scrollbarSize		设置滚动条的宽度
android:scrollbarStyle		设置滚动条的风格和位置。设置值：insideOverlay、insideInset、outsideOverlay、outsideInset
android:scrollbars		设置滚动条显示。none（隐藏），horizontal（水平），vertical（垂直）。使用该属性让 EditText 内有滚动条。但是其他容器如 LinearLayout 设置了但是没有效果
android:tag		设置一个文本标签。可以通过 View.getTag()或 View.findViewWithTag()检索含有该标签字符串的 View。但一般最好通过 ID 来查询 View，因为速度更快，并且允许编译时类型检查
android:visibility	setVisibility(int)	设置是否显示 View。设置值：visible（默认值，显示）、invisible（不显示，但是仍然占用空间）、gone（不显示，不占用空间）

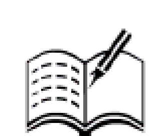
4.2　线性布局

线性布局（LinearLayout）是较简单的一个布局，它提供了控件水平或者垂直排列的模型。本节将会对线性布局进行简单介绍，首先介绍 LinearLayout 类的相关知识，然后通过一个实例说明 LinearLayout 的使用方法。

4.2.1　LinearLayout 类简介

LinearLayout 通过设置的垂直或水平的属性值，来排列所有的子元素。所有的子元素





都被堆放在其他元素之后，因此一个垂直列表的每一行只会有一个元素，而不管它们有多宽，而一个水平列表将会只有一个行高（高度为最高子元素的高度加上边框高度）。LinearLayout 保持子元素之间的间隔以及互相对齐（相对一个元素的右对齐、中间对齐或者左对齐）。

LinearLayout 的常用属性及对应设置方法如表 4-2 所示。

表 4-2 LinearLayout 的常用属性及对应设置方法

属性名称	设置方法	描述
android:orientation	setOrientation(int)	设置线性布局的朝向，可设置为 horizontal、vertical 两种排列方式
android:gravity	setGravity(int)	设置线性布局的内部元素的对齐方式

1. orientation 属性

在线性布局中可以使用 orientation 属性来设置布局的朝向，可取值及说明如下。

- horizontal：定义横向布局。
- vertical：定义纵向布局。

对于纵向布局与横向布局而言，控件的排列方式分别如图 4-2 和图 4-3 所示。

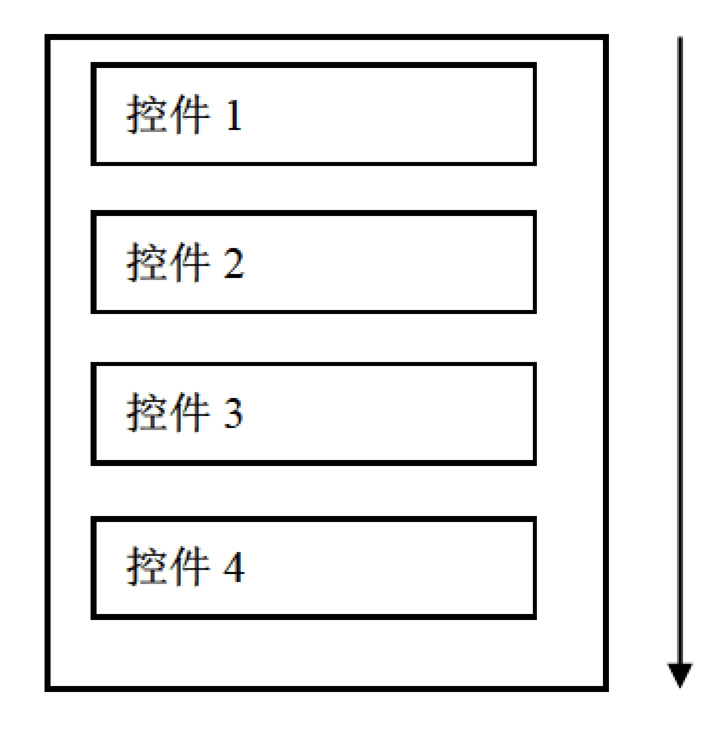


图 4-2 纵向布局

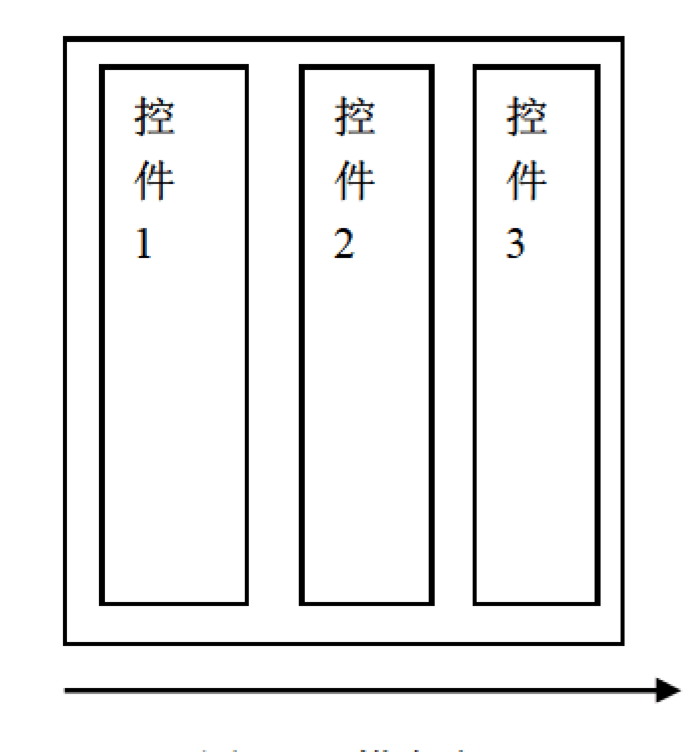


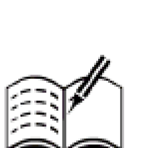
图 4-3 横向布局

2. gravity 属性

在线性布局中可以使用 gravity 属性设置控件的对齐方式，可取的值及说明如表 4-3 所示。

表 4-3 gravity 属性

常量	描述
top	不改变控件大小，对齐到容器顶部
bottom	不改变控件大小，对齐到容器底部
left	不改变控件大小，对齐到容器左侧
right	不改变控件大小，对齐到容器右侧
center_vertical	不改变控件大小，对齐到容器纵向中央位置
fill_vertical	纵向拉伸以填充满容器



Note

续表

常　　量	描　　述
center_horizontal	不改变控件大小，对齐到容器横向中央位置
fill_horizontal	横向拉伸以填充满容器
center	不改变控件大小，放置在容器的正中间
fill	横向和纵向同时拉伸以填充满容器

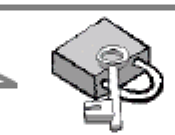
4.2.2　线性布局实例

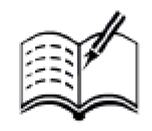
本节将通过一个实例来说明 LinearLayout 的使用方法。在本实例中，最上层的纵向线性布局中嵌套了一个纵向线性布局和一个横向线性布局。在嵌套的纵向线性布局中，摆放了一个 TextView 和一个 Button 控件；在嵌套的横向线性布局中摆放了两个 TextView 控件。本实例开发步骤如下：

（1）创建项目 EX04_1。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="本案例演示 LinearLayout 线性布局"
            android:textSize="20px"
            />
     <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
         android:orientation="vertical"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         >
         <TextView
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="这是纵向布局的第一个 TextView。"
             android:textSize="20px"
             />
         <Button
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:layout_gravity="right"
             android:text="这是一个按钮"
```






```
29                  />
30          </LinearLayout>
31          <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
32              android:orientation="horizontal"
33              android:layout_width="fill_parent"
34              android:layout_height="fill_parent"
35              >
36              <TextView
37                  android:layout_width="wrap_content"
38                  android:layout_height="wrap_content"
39                  android:text="第一个 TextView"
40                  android:textSize="20px"
41                  android:padding="2px"
42                  />
43              <TextView
44                  android:layout_width="wrap_content"
45                  android:layout_height="wrap_content"
46                  android:text="第二个 TextView"
47                  android:textSize="20px"
48                  android:padding="2px"
49                  />
50          </LinearLayout>
51 </LinearLayout>
```

说明：

- 第 2~6 行：第 3 行代码声明该布局为一个纵向布局；第 4~5 行代码设置该布局高度和宽度填充满整个容器。对于最顶层的布局来说，它的容器就是手机屏幕，所以该布局会填充手机屏幕进行显示。
- 第 7~12 行：在最顶层的布局中声明第一个控件 TextView。第 9 行代码定义 TextView 控件的高度，wrap_content 的含义是根据视图内部内容自动扩展以适应其大小；第 11 行代码定义 TextView 控件的字体大小为 20px。
- 第 13~30 行：在最顶层的布局中嵌套一个纵向线性布局。第 14 行代码定义该布局的朝向为纵向。在该布局中包含一个 TextView 控件与一个 Button 控件；第 27 行代码定义 Button 控件的对齐方式为右对齐（即 Button 放在该布局的最右侧）。
- 第 31~50 行：在最顶层的布局中嵌套一个横向线性布局。第 32 行代码定义该布局的朝向为横向；第 34 行代码定义该布局填充满顶层布局的剩余空间。在该布局中包含两个 TextView 控件；第 41 行代码定义 TextView 的内容与父容器边界的距离为 2px（2 个像素）。android:padding 规定父 view 里面的内容与父 view 边界的距离。

本实例运行结果如图 4-4 所示。



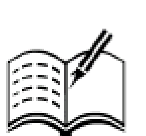




图 4-4 EX04_1 运行结果

4.3 表格布局

表格布局（TableLayout）是按照行列来组织子视图的布局，包含一系列的 TableRow 对象，用于定义行。本节将会对表格布局进行介绍，首先介绍 TableLayout 类的相关知识，然后通过一个实例说明 TableLayout 的使用方法。

4.3.1 TableLayout 类简介

表格布局包含一系列的 TableRow 对象，用于定义行。表格布局不为它的行、列和单元格显示表格线。每个行可以包含 0 个以上（包括 0）的单元格；每个单元格可以设置一个 View 对象。与行包含很多单元格一样，表格包含很多列。表格的单元格既可以为空，也可以像 HTML 那样跨列。

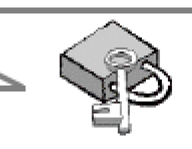
无论是在代码还是在 XML 布局文件中，单元格必须按照索引顺序加入表格行。列号从 0 开始，如果不为子单元格指定列号，其将自动增值，使用下一个可用列号。虽然表格布局典型的子对象是表格行，但实际上可以使用任何视图类的子类，作为表格视图的直接子对象，视图会作为一行并合并了所有列的单元格显示。

列的宽度由该列所有行中最宽的一个单元格决定，而表格的总宽度由其父容器决定。不过表格布局可以通过 setColumnShrinkable()或者 setColumnStretchable()方法来标记哪些列可以收缩或拉伸。如果标记为可以收缩，列宽可以收缩以使表格适合容器的大小；如果标记为可以拉伸，列宽可以拉伸以占用多余的空间。可以通过调用 setColumnCollapsed()方法来隐藏列。

在表格布局中，可以为列设置以下 3 种属性。

（1）Shrinkable：表示列的宽度可以进行收缩，以使表格能够适应其父容器的大小。

（2）Stretchable：表示列的宽度可以进行拉伸，以填满表格中空闲的空间。

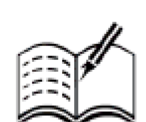


（3）Collapsed：表示列将会被隐藏。



注意

列可以同时具有可拉伸和可收缩标记，这一点是很重要的，这种情况下，该列的宽度将任意拉伸或收缩以适应父容器。



Note

从图 4-1 中可以看到，TableLayout 继承自 LinearLayout 类，除了继承 LinearLayout 类的属性和方法外，TableLayout 类中还包含表格布局自身的属性和方法。TableLayout 的常用属性及对应设置方法如表 4-4 所示。

表 4-4　TableLayout 的常用属性及对应设置方法

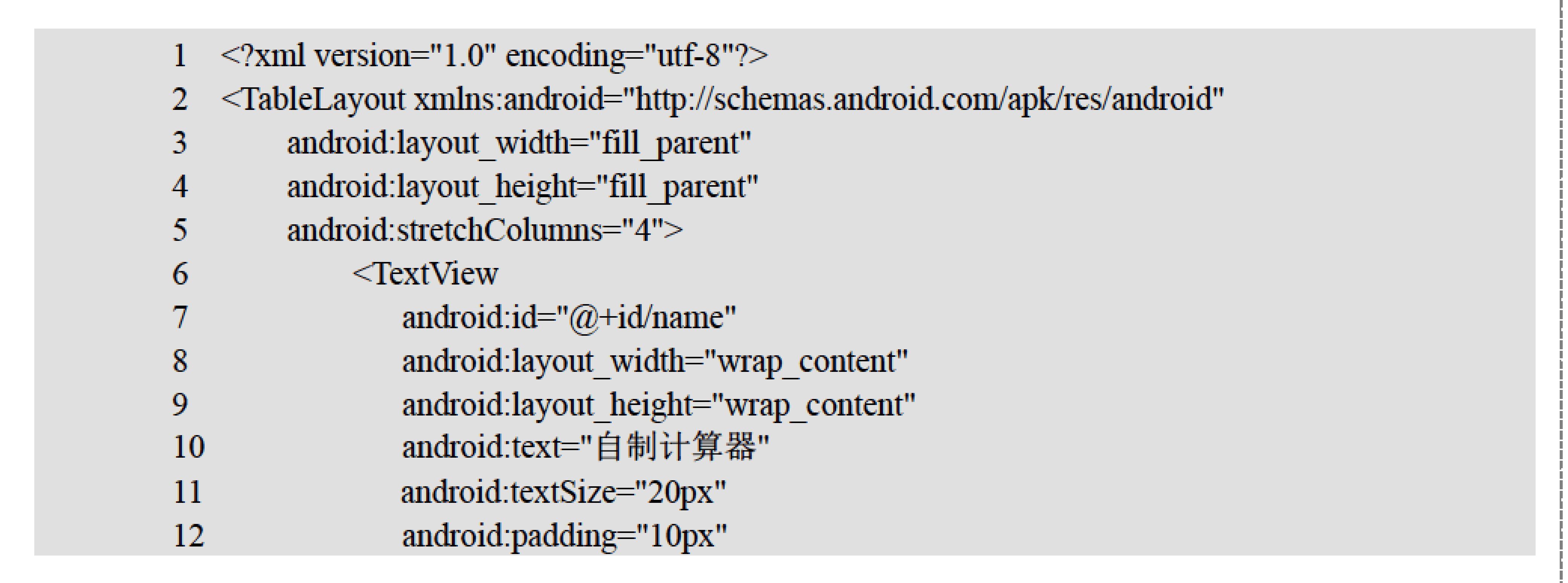
属 性 名 称	相 关 方 法	描　述
android:collapseColumns	setColumnCollapsed(int,boolean)	隐藏从 0 开始的索引列。列号必须用逗号隔开，如 1, 2, …。非法或重复的设置将被忽略
android:shrinkColumns	setShrinkAllColumns(boolean)	收缩从 0 开始的索引列。列号必须用逗号隔开，如 1, 2, …。非法或重复的设置将被忽略。可以通过“*”代替收缩所有列。注意一列能同时表示收缩和拉伸
android:stretchColumns	setStretchAllColumns(boolean)	拉伸从 0 开始的索引列。列号必须用逗号隔开，如 1, 2, …。非法或重复的设置将被忽略。可以通过“*”代替拉伸所有列。注意一列能同时表示收缩和拉伸

4.3.2　表格布局实例

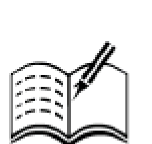
本节将通过一个实例来说明 TableLayout 的使用方法。在本实例中，实现一个计算器的界面。本实例开发步骤如下：

（1）创建项目 EX04_2。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

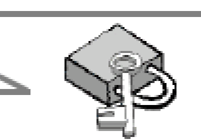


```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="4">
        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="自制计算器"
            android:textSize="20px"
            android:padding="10px"
```

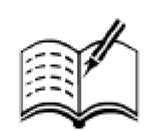


Note

```
            />
             <TextView
                android:layout_height="50px"
                android:textSize="20px"
                android:gravity="right"
                android:background="#FFFFFF"
            />
        <TableRow android:paddingTop="20px">
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="20px"
                android:textSize="20px"
                android:text="1"
            />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="20px"
                android:textSize="20px"
                android:text="2"
            />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="20px"
                android:textSize="20px"
                android:text="3"
            />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="20px"
                android:textSize="20px"
                android:text="&lt;--"
                />
                <Button
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:text="+"
                />
          </TableRow>
          <TableRow>
                ...
           </TableRow>
          <TableRow>
                ...
           </TableRow>
          <TableRow>
```



```
62                 …
63             </TableRow>
64     </TableLayout>
```



说明：

- 第 2~5 行：定义一个表格布局。第 3~4 行代码定义表格布局布满整个屏幕；第 5 行代码定义该表格布局第 5 列是可拉伸的，以布满整个表格。
- 第 6~13 行：在表格布局中定义第一个 TextView 控件。第 12 行代码定义该 TextView 控件中的文字距离边框的距离为 20px。
- 第 14~19 行：在表格布局中定义第二个 TextView 控件。第 18 行代码定义该 TextView 控件的背景颜色为白色。
- 第 20~54 行：定义一个 TableRow，表示表格布局的一行。在该行中有 5 个 Button，分别显示 1、2、3、<--、+。第 20 行代码定义该 TableRow 的上边距为 20px；第 21~27 行代码定义该行的第一个 Button；第 28~34 行代码定义该行的第 2 个 Button，第 35~41 行代码定义该行的第 3 个 Button；第 42~48 行代码定义该行的第 4 个 Button；第 49~53 行代码定义该行的第 5 个 Button。
- 第 55~57 行、58~60 行、61~63 行代码分别定义该表格布局的其余 3 个 TableRow。

本实例运行结果如图 4-5 所示。



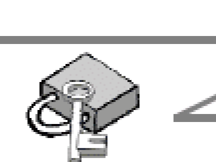
图 4-5　EX04_2 运行结果

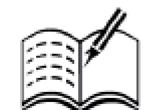
4.4　相对布局

相对布局（RelativeLayout）是指在容器内部的子元素可以使用彼此之间的相对位置或者和容器间的相对位置来进行定位。本节将会对相对布局进行介绍，首先介绍 RelativeLayout 类的相关知识，然后通过一个实例说明 RelativeLayout 的使用方法。

4.4.1　RelativeLayout 类简介

在相对布局中，控件的位置是相对其他控件或者父容器而言的。在进行设计时，需要





Note

按照控件之间的依赖关系排列，例如，控件 B 的位置相对于控件 A 决定，则在布局文件中控件 A 需要在控件 B 的前面进行定义。

在设计相对布局时，会用到很多属性，下面对属性分别进行说明，如表 4-5 所示。

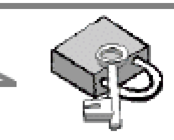
表 4-5　RalativeLayout 属性

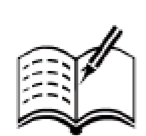
属　　性	值	描　　述
android:layout_alignParentTop	true 或 false	如果为 true，该控件的顶部与其父控件的顶部对齐
android:layout_alignParentBottom	true 或 false	如果为 true，该控件的底部与其父控件的底部对齐
android:layout_alignParentLeft	true 或 false	如果为 true，该控件的左部与其父控件的左部对齐
android:layout_alignParentRight	true 或 false	如果为 true，该控件的右部与其父控件的右部对齐
android:layout_alignWithParentIfMissing	true 或 false	参考控件不存在或不可见时参照父控件
android:layout_centerHorizontal	true 或 false	如果为 true，该控件置于父控件的水平居中位置
android:layout_centerVertical	true 或 false	如果为 true，该控件置于父控件的垂直居中位置
android:layout_centerInParent	true 或 false	如果为 true，该控件置于父控件的中央位置
android:layout_above	某控件的 id 属性	将该控件的底部置于给定 ID 控件的上方
android:layout_below	某控件的 id 属性	将该控件的底部置于给定 ID 控件的下方
android:layout_toLeftOf	某控件的 id 属性	将该控件的右边缘与给定 ID 的控件左边缘对齐
android:layout_toRightOf	某控件的 id 属性	将该控件的左边缘与给定 ID 的控件右边缘对齐
android:layout_alignBaseline	某控件的 id 属性	将该控件的 baseline 与给定 ID 的 baseline 对齐
android:layout_alignTop	某控件的 id 属性	将该控件的顶部边缘与给定 ID 的顶部边缘对齐
android:layout_alignBottom	某控件的 id 属性	将该控件的底部边缘与给定 ID 的底部边缘对齐
android:layout_alignLeft	某控件的 id 属性	将该控件的左边缘与给定 ID 的左边缘对齐
android:layout_alignRight	某控件的 id 属性	将该控件的右边缘与给定 ID 的右边缘对齐
android:layout_marginTop	int 类型数值	上偏移的值
android:layout_marginBottom	int 类型数值	下偏移的值
android:layout_marginLeft	int 类型数值	左偏移的值
android:layout_marginRight	int 类型数值	右偏移的值



注意

能在 RelativeLayout 容器本身及其子元素之间产生循环依赖。例如，不能将 RelativeLayout 的高设置为 WRAP_CONTENT 时，将子元素的高设置为 ALIGN_PARENT_BOTTOM。





Note

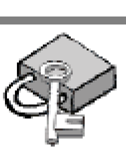
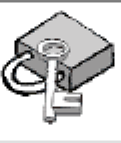
4.4.2　相对布局实例

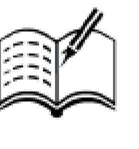
本节将用一个实例来说明相对布局的使用方法。在本实例中，采用相对布局来实现计算器的界面。本实例开发步骤如下：

（1）创建项目 EX04_3。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="自制计算器"
        android:textSize="20px"
        android:padding="5px"
        />
    <TextView
        android:id="@+id/expr"
        android:layout_width="fill_parent"
        android:layout_height="50px"
        android:layout_below="@id/name"
        android:background="#FFFFFF"
        android:textSize="40px"
        android:gravity="right"
        />
    <Button
        android:id="@+id/num1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/expr"
        android:layout_alignLeft="@id/expr"
        android:padding="20px"
        android:textSize="20px"
        android:text="1"
        />
    <Button
        android:id="@+id/num2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/expr"
        android:layout_toRightOf="@id/num1"
        android:padding="20px"
        android:textSize="20px"
```

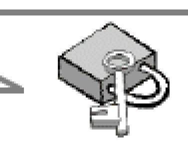


Note

```
42                android:text="2"
43                />
44          <Button
45                android:id="@+id/num3"
46                android:layout_width="wrap_content"
47                android:layout_height="wrap_content"
48                android:layout_below="@id/expr"
49                android:layout_toRightOf="@id/num2"
50                android:padding="20px"
51                android:textSize="20px"
52                android:text="3"
53                />
54          <Button
55                android:id="@+id/back"
56                android:layout_width="wrap_content"
57                android:layout_height="wrap_content"
58                android:layout_toRightOf="@id/num3"
59                android:layout_alignTop="@id/num3"
60                android:padding="20px"
61                android:textSize="20px"
62                android:text="&lt;--"
63                />
64           <Button
65               android:id="@+id/add"
66                android:layout_width="fill_parent"
67                android:layout_height="wrap_content"
68                android:layout_toRightOf="@id/back"
69                android:layout_alignTop="@id/back"
70                android:padding="20px"
71                android:textSize="20px"
72                android:text="+"
73                />
74      …
75      </RelativeLayout>
```

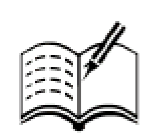
说明：

- 第 2~6 行：定义一个相对布局，大小充满整个屏幕。
- 第 7~14 行：定义一个 ID 为 name 的 TextView 控件。第 8 行代码定义该 TextView 的 ID 为 name。
- 第 15~23 行：定义一个 ID 为 expr 的 TextView 控件。第 16 行代码定义该 TextView 的 ID 为 expr；第 18 行代码定义该控件高度为 50px；第 19 行代码定义该控件位于 ID 为 name 的控件的下方；第 21 行代码定义该控件的文字大小为 40px；第 22 行代码定义该控件的对齐方式为右对齐。
- 第 24~33 行：定义一个 ID 为 num1 的 Button 控件。第 25 行代码定义该 Button 的 ID 为 num1；第 28 行代码定义该控件位于 ID 为 expr 的控件的下方；第 29 行代码定义该控件的左边缘与 ID 为 expr 的控件的左边缘对齐；第 32 行代码定义该



控件的内容为 1。

- 第 34~43 行：定义一个 ID 为 num2 的 Button 控件。第 35 行代码定义该 Button 的 ID 为 num2；第 38 行代码定义该控件位于 ID 为 expr 的控件的下方；第 39 行代码定义该控件位于 ID 为 num1 的控件的右侧；第 42 行代码定义该控件的内容为 2。
- 第 44~53 行：定义一个 ID 为 num3 的 Button 控件。第 45 行代码定义该 Button 的 ID 为 num3；第 48 行代码定义该控件位于 ID 为 expr 的控件的下方；第 49 行代码定义该控件位于 ID 为 num2 的控件的右侧；第 52 行代码定义该控件的内容为 3。
- 第 54~63 行：定义一个 ID 为 back 的 Button 控件。第 55 行代码定义该 Button 的 ID 为 back；第 58 行代码定义该控件位于 ID 为 num3 的控件的右侧；第 59 行代码定义该控件的上边缘与 ID 为 num3 的控件的上边缘对齐；第 62 行代码定义该控件的内容为<--，显示为<--。
- 第 64~73 行：定义一个 ID 为 add 的 Button 控件。第 65 行代码定义该 Button 的 ID 为 add；第 66 行代码定义该控件的宽度为填充满父控件剩余的空间；第 68 行代码定义该控件位于 ID 为 back 的控件的右侧；第 69 行代码定义该控件的上边缘与 ID 为 back 的控件的上边缘对齐；第 72 行代码定义该控件的内容为+。
- 其余行的代码与上述代码相似，此处不再详细介绍。



Note

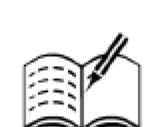
本实例运行结果如图 4-6 所示。



图 4-6　EX04_3 运行结果

4.5 帧 布 局

帧布局（FrameLayout）是五大布局中最简单的一个布局，在这个布局中，整个界面被当成一块空白备用区域，所有的子元素都不能指定位置进行放置，它们全部放于该区域的左上角，并且后面的子元素直接覆盖在前面的子元素之上，将前面的子元素部分或全部遮挡。本节将对帧布局进行介绍，首先介绍 FrameLayout 类的相关知识，然后通过一个实例说明 FrameLayout 的使用方法。



Note

4.5.1　FrameLayout 类简介

帧布局把屏幕当作一块区域，在这块区域中可以添加多个子控件，但是所有的子控件都被对齐到屏幕的左上角。帧布局的大小由子控件中尺寸最大的控件来决定。

FrameLayout 类的常用属性及对应设置方法如表 4-6 所示：

表 4-6　FrameLayout 类的常用属性及对应设置方法

属　　性	对 应 方 法	描　　述
android:foreground	setForeground(Drawable)	设置绘制在所有子控件之上的内容
android:forefroundGravity	SetForeground(int)	设置绘制在所有子控件之上内容的对齐方式

4.5.2　帧布局实例

本节将通过一个实例来说明 FrameLayout 的使用方法。本实例开发步骤如下：

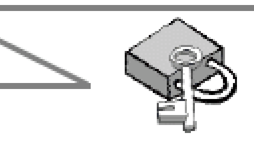
（1）创建项目 EX04_4。

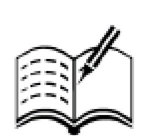
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="fill_parent"
4       android:layout_height="fill_parent"
5       >
6       <TextView
7           android:layout_width="fill_parent"
8           android:layout_height="wrap_content"
9           android:text="这是第一个 TextView"
10          />
11      <TextView
12          android:layout_width="fill_parent"
13          android:layout_height="wrap_content"
14          android:text="这是第二个 TextView"
15          android:textSize="20px"
16          android:gravity="right"
17          />
18      <TextView
19          android:layout_width="wrap_content"
20          android:layout_height="50px"
21          android:text="这是第三个 TextView"
22          android:textSize="30px"
23          />
24  </FrameLayout>
```

说明：

- 第 2~5 行：定义一个帧布局。该布局大小充满整个手机屏幕。
- 第 6~10 行：定义一个 TextView 控件。





Note

- 第 11~17 行：定义一个 TextView 控件。第 15 行代码定义该 Textview 的字体大小为 20px；第 16 行代码定义该 TextView 的对齐方式为右对齐。
- 第 18~23 行：定义一个 TextView 控件。

本实例运行结果如图 4-7 所示。



图 4-7　EX04_4 运行结果

4.6 绝对布局

绝对布局（AbsoluteLayout）是指所有控件的排列由开发人员通过控件的坐标来指定，容器不再负责管理其子控件的位置。本节将对绝对布局进行介绍，首先介绍 AbsoluteLayout 类的相关知识，然后通过一个实例说明 AbsoluteLayout 的使用方法。

4.6.1　AbsoluteLayout 类简介

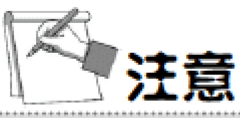
在 AbsoluteLayout 中，由于子控件的位置和布局都通过坐标来指定，所以在设计布局时，开发人员需要指定子元素精确的横坐标和纵坐标。因此，AbsoluteLayout 类中并没有特有的属性和方法。

绝对布局缺乏灵活性，在没有绝对定位的情况下相比其他类型的布局更难维护，并且采用绝对布局设计的界面有可能在不同的手机设备上显示完全不同的结果。因此，在选择设计布局时，不推荐使用绝对布局。

AbsoluteLayout 类的常用属性及对应设置方法如表 4-7 所示。

表 4-7　AbsoluteLayout 类的常用属性及对应设置方法

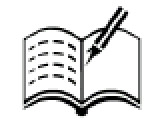
属　性	描　述
android:layout_x	指定控件的 x 坐标
android:layout_y	指定控件的 y 坐标



注意

对于手机屏幕而言，坐标原点为屏幕左上角。当向右或者向下移动时，坐标值将变大。

4.6.2 绝对布局实例



Note

本节将通过一个实例来说明 AbsoluteLayout 的使用方法。本实例开发步骤如下：

（1）创建项目 EX04_5。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       >
7       <EditText
8           android:text="本案例演示绝对布局"
9           android:layout_width="fill_parent"
10          android:layout_height="wrap_content"
11          />
12      <Button
13          android:layout x="250px"
14          android:layout y="50px"
15          android:layout width="70px"
16          android:layout height="wrap content"
17          android:text="Button"
18          />
19  </AbsoluteLayout>
```

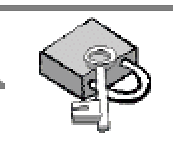
说明：

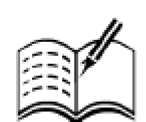
- ❑ 第 2~6 行：定义一个绝对布局。该布局大小充满整个手机屏幕。
- ❑ 第 7~11 行：定义一个 EditText 控件。
- ❑ 第 12~18 行：定义一个 Button 控件。第 13 行代码定义该控件的横坐标为 250px；第 14 行代码定义该控件的纵坐标为 50px；第 15 行代码定义该控件的宽度为 70px。

本实例运行结果如图 4-8 所示。



图 4-8　EX04_5 运行结果





4.7 布局的嵌套

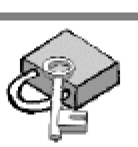
前面讲述了 Android 的五大布局，在进行 Android 应用程序的界面设计时，开发人员可以根据界面的需要选择相应布局。此外，Android 的五大布局还可以进行相互的嵌套，来满足界面的设计要求。

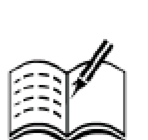
本节将用一个实例来说明布局的嵌套使用方法。在本实例中，采用布局之间相互嵌套的方法实现计算器的界面。本实例的开发步骤如下：

（1）创建项目 EX04_6。

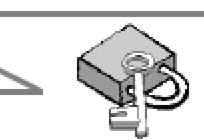
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

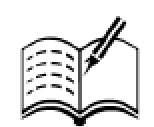
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="本案例演示布局的嵌套"
        android:textSize="20px"
     />
     <TextView
        android:layout_width="fill_parent"
        android:layout_height="50px"
        android:textSize="40px"
        android:gravity="right"
        android:background="#FFFFFF"
     />
     <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
     >
        <Button
            android:id="@+id/num1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:padding="20px"
            android:textSize="20px"
            android:text="1"
        />
        <Button
            android:id="@+id/num2"
            android:layout_width="wrap_content"
```






```
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/num1"
            android:padding="20px"
            android:textSize="20px"
            android:text="2"
        />
        <Button
            android:id="@+id/num3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/num2"
            android:padding="20px"
            android:textSize="20px"
            android:text="3"
            />
        <Button
            android:id="@+id/back"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/num3"
            android:padding="20px"
            android:textSize="20px"
            android:text="BACK"
            />
        <Button
            android:id="@+id/add"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/back"
            android:padding="20px"
            android:textSize="20px"
            android:text="+"
            />
    </RelativeLayout>
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    >
            …
    </RelativeLayout>
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    >
        …
    </RelativeLayout>
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
```





Note

```
176                 android:layout_width="fill_parent"
177                 android:layout_height="wrap_content"
178         >
179                 …
227         </RelativeLayout>
228     </LinearLayout>
```

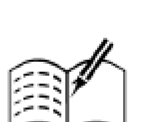
说明：

- 第 2~6 行：定义一个线性布局。第 3 行代码定义该线性布局为纵向布局；第 4、5 行代码定义该线性布局布满整个手机屏幕。
- 第 7~12 行：定义一个 TextView 控件。
- 第 13~19 行：定义一个 TextView 控件。第 15 行代码定义该控件的高度为 50px；第 16 行代码定义该控件的文本大小为 40px；第 17 行代码定义该控件的对齐方式为右对齐；第 18 行代码定义该控件的背景颜色为白色。
- 第 20~70 行：在顶层的线性布局中嵌套一个相对布局。
  - 第 21 行：定义该相对布局的朝向为横向。
  - 第 25~33 行：在相对布局中定义一个 Button 控件。第 26 行代码定义该 Button 的 ID 为 num1；第 29 行代码定义该控件的左边缘与父控件的左边缘对齐。
  - 第 34~42 行：在相对布局中定义一个 Button 控件。第 35 行代码定义该 Button 的 ID 为 num2；第 38 行代码定义该控件位于 ID 为 num1 的控件右侧。
  - 第 43~51 行：在相对布局中定义一个 Button 控件。第 44 行代码定义该 Button 的 ID 为 num3；第 47 行代码定义该控件位于 ID 为 num2 的控件右侧。
  - 第 52~60 行：在相对布局中定义一个 Button 控件。第 53 行代码定义该 Button 的 ID 为 back；第 56 行代码定义该控件位于 ID 为 num3 的控件右侧。
  - 第 61~69 行：在相对布局中定义一个 Button 控件。第 62 行代码定义该 Button 的 ID 为 add；第 65 行代码定义该控件位于 ID 为 back 的控件右侧。
- 第 71~121 行、第 122~173 行、第 174~227 行分别定义其他 3 个相对布局，这 3 个相对布局均嵌套在顶层的线性布局中。每一个相对布局的代码与第一个相对布局的代码相似，这里不再详细介绍，详情见本实例代码。

本实例运行结果如图 4-9 所示。



图 4-9　EX04_6 运行结果


Note

4.8 习　　题

1．简述 Android 中常用的 5 种布局方式。

2．简述 View 类。

3．在 Android 项目中使用线性布局方式实现如图 4-10 和图 4-11 所示的界面。



图 4-10　纵向线性布局界面



图 4-11　横向线性布局界面

4．在 Android 项目中使用表格布局方式实现如图 4-12 所示的界面。

5．在 Android 项目中使用相对布局方式实现如图 4-12 所示的界面。

6．在 Android 项目中使用帧布局方式实现如图 4-13 所示的界面。



图 4-12　表格布局界面



图 4-13　帧布局界面

7．在 Android 项目中使用布局相互嵌套方式设计简单运算器的界面，并实现该运算器程序。如图 4-14 所示，在该程序中输入运算数字，然后单击下面的运算符，再单击“计算”按钮，得到运算结果界面。



图 4-14　运算器界面

第 5 章

常用基本控件

【本章内容】

- ❑ 文本控件
- ❑ 按钮控件
- ❑ 单选按钮
- ❑ 复选框
- ❑ 图片控件
- ❑ 时钟控件
- ❑ 日期与时间控件

应用程序的界面由很多控件构成，Android 应用程序同样如此。Android 平台提供了许多简单、易用的控件。本章将对常用的 Android 基本控件进行介绍。

5.1　文 本 控 件

在 Android 中，文本控件主要包括 TextView 和 EditText 两种，本节将对这两种控件的用法进行详细介绍。

5.1.1　TextView 类简介

TextView 类的继承关系如图 5-1 所示，其主要功能是向用户显示文本内容。一个 TextView 其实是一个文本编辑器，只不过被设置为不允许编辑，而其子类 EditText 被设置为允许用户对内容进行编辑。

```
java.lang.Object
 └android.view.View
    └android.widget.TextView
```

图 5-1　TextView 类继承关系

TextView 控件中包含许多属性，这些属性可以在 XML 文件中设置，也可以在代码中动态声明。TextView 常用属性及对应方法如表 5-1 所示。

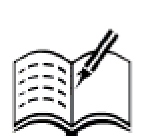
表 5-1　TextView 常用属性及对应方法

属 性 名 称	对 应 方 法	说　　明
android:autoLink	setAutoLinkMask(int)	设置是否将指定格式的文本转换为可单击的超链接提示。其值可取 web、email、phone、map、all
android:gravity	setGravity(int)	定义 TextView 在 x 轴和 y 轴方向上的显示方式



续表

属性名称	对应方法	说明
android:height	setHeight(int)	定义 TextView 的准确高度，以像素为单位
android:width	setWidth(int)	定义 TextView 的宽度，以像素为单位
android:hint	setHint(int)	当 TextView 中显示的内容为空时，显示该文本
android:text	setText(CharSequence)	设置 TextView 显示的内容
android:textColor	setTextColor(ColorStateList)	设置 TextView 的文本颜色
android:textSize	setTextSize(float)	设置 TextView 的文本大小
android:ellipsize	setEllipsize(TextUtils.TruncateAt)	如果设置了该属性，当 TextView 中要显示的内容超过了 TextView 的长度时，会对内容进行省略。其值可取 start、middle、end、marquee


Note

5.1.2 EditText 类简介

EditText 类的继承关系如图 5-2 所示，用户可以对 EditText 控件进行编辑，同时还可以为 EditText 控件设置监听器，用来检测用户输入是否合法等。EditText 常用属性及对应方法如表 5-2 所示。

```
java.lang.Object
  └android.view.View
      └android.widget.TextView
          └android.widget.EditText
```

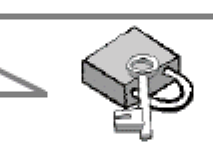
图 5-2 EditText 类继承关系

表 5-2 EditText 常用属性及对应方法

属性名称	对应方法	说明
android:cursorVisible	setCursorVisible(boolean)	设置光标是否可见，默认可见
android:lines	setLines(int)	通过设置固定的行数来决定 EditText 的高度
android:maxLines	setMaxLines(int)	设置最大的行数
android:minLines	setMinLines(int)	设置最小的行数
android:password	setTransformationMethod (TransformationMethod)	设置文本框中的内容是否显示为密码
android:phoneNumber	setKeyListener(KeyListener)	设置文本框中的内容只能是电话号码
android:scrollHorizontally	setHorizontallyScrolling(boolean)	设置文本框是否可以进行水平滚动
android:singleLine	setTransformationMethod (TransformationMethod)	设置文本框为单行模式
android:maxLength	setFilters(InputFilter)	设置最大显示长度
android:textStyle	setTypeface(Typeface)	设置字形：bold（粗体）0、italic（斜体）1、bolditalic（又粗又斜）2，可以设置一个或多个字形，用“|”隔开

5.1.3 文本控件使用实例

本节将通过一个实例来介绍文本控件的使用方法。本实例所完成的功能比较简单，在

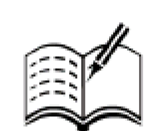


用户没有任何输入时 EditText 的默认显示为“请输入 E-mail”，TextView 的显示为空，而当用户输入数据后，程序将用户输入到 EditText 中的数据实时显示到 TextView 中。

本实例的开发步骤如下：

（1）新建项目 EX05_1。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：



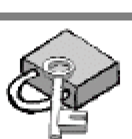

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6         >
7     <EditText
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:id="@+id/editText1"
11     android:hint="请输入 E-mail"
12     />
13     <TextView
14     android:layout_width="fill_parent"
15     android:layout_height="wrap_content"
16     android:textSize="16sp"
17     android:text=""
18     android:id="@+id/textView1"
19     />
20     </LinearLayout>
```

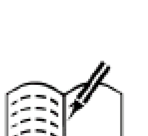
说明：

- 第 7~12 行：声明一个 EditText 控件。
  - 第 8 行：其意义为宽度填充父组件。
  - 第 9 行：其意义为高度随内容自适应。
  - 第 10 行：声明此 EditText 控件的 ID 为 editText1。
  - 第 11 行：在 EditText 的内容为空时显示“请输入 E-mail”来提醒用户。
- 第 13~19 行：声明一个 TextView 控件。
  - 第 14 行：其意义为宽度填充父组件。
  - 第 15 行：其意义为高度随内容自适应。
  - 第 16 行：设置 TextView 控件的 android:textSize="16sp"，其意义为字体大小为 16sp。
  - 第 17 行：意义为在默认情况下，TextView 的显示为空。
  - 第 18 行：声明此 TextView 控件的 ID 为 textView1。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，输入电子信箱地址，然后显示在 TextView 控件中。编写代码如下：

```
package wyq.EX05_1;
```





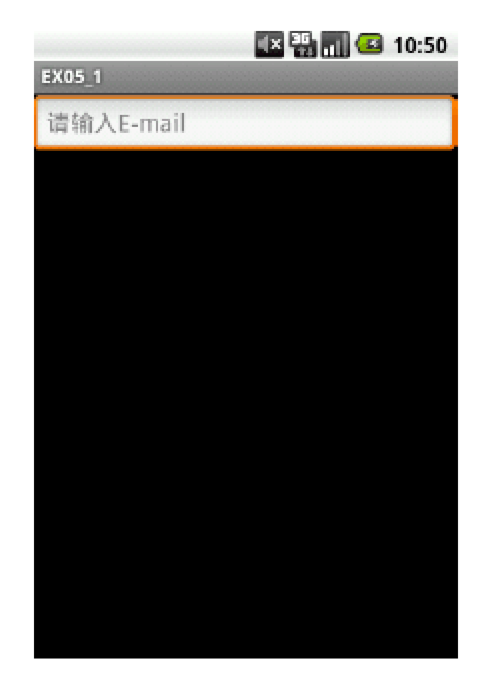
Note

```
2    import android.app.Activity;
3    import android.os.Bundle;
4    import android.view.KeyEvent;
5    import android.view.View;
6    import android.widget.EditText;
7    import android.widget.TextView;
8    public class FirstActivity extends Activity {
9            EditText et;
10           TextView tv;
11           public void onCreate(Bundle savedInstanceState) {
12               super.onCreate(savedInstanceState);
13               setContentView(R.layout.main);
14               et=(EditText) findViewById(R.id.editText1);
15               tv=(TextView) findViewById(R.id.textView1);
16               et.setOnKeyListener(new EditText.OnKeyListener()
17               {
18                public boolean onKey(View arg0, int arg1, KeyEvent arg2)
19                      {
20                           tv.setText(et.getText());
21                           return false;
22                      }
23               });
24           }
25   }
```

说明：

- 第 9~10 行：分别声明了一个 EditText 控件和一个 TextView 控件。
- 第 14~15 行：通过 findViewById 分别获取 main.xml 中声明的 EditText 控件和 TextView 控件。
- 第 16 行：为 EditText 添加了一个 setOnKeyListener 监听事件，并且在第 18 行设置了 onKey()方法，即在有键盘操作时触发此事件。
- 第 20 行：为 onKey 事件的处理过程，在此处即为实时获取用户输入到 EditText 中的数据并显示在 TextView 中。

本实例运行结果如图 5-3 所示。



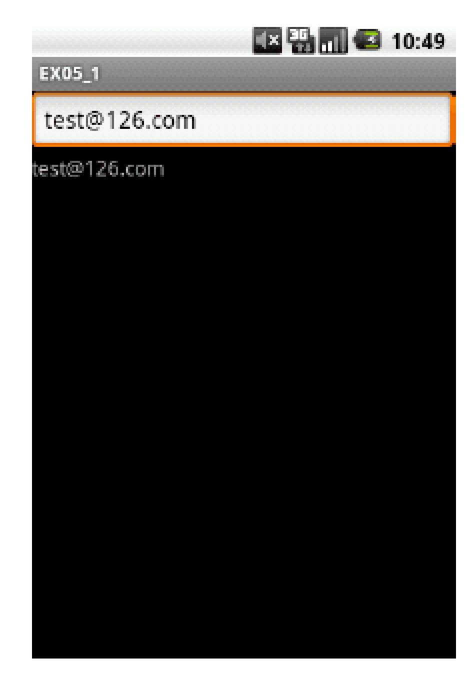
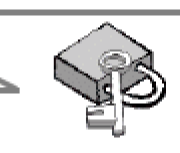
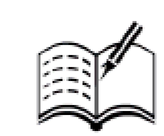


图 5-3　EX05_1 运行结果





5.2 按钮控件

Android 中的按钮控件主要包括 Button 控件和 ImageButton 控件。通过为按钮控件增加监听事件来产生相应的命令，完成某一个功能。本节将对这两种控件进行详细介绍。

5.2.1 Button 类简介

Button 类的继承关系如图 5-4 所示。用户可以对 Button 控件执行按下或者单击等操作来完成某项功能。Button 控件的用法主要是为 Button 控件设置 View.OnClickListener 监听器并在监听器的实现代码中开发按钮按下事件的处理代码：

```
java.lang.Object
  └android.view.View
      └android.widget.TextView
          └android.widget.Button
```

图 5-4　Button 类继承关系

```
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            // 处理过程
    }
});
```

另一种方法是在 XML 布局文件中通过 Button 的 android:onClick 属性指定一个方法：

```
android:onClick="selfDestruct"
```

以替代在 activity 中为 Button 设置 OnClickListener，但是为了正确执行，该方法必须声明为 public 并且仅接受一个 View 类型的参数：

```
public void selfDestruct(View view) {
    // 处理过程
}
```

5.2.2 ImageButton 类简介

ImageButton 类的继承关系如图 5-5 所示。ImageButton 控件与 Button 控件的主要区别是 ImageButton 中没有 text 属性，即按钮中显示图片而不是文本。ImageButton 控件中设置按钮显示的图片可以通过 android:src 属性来实现：

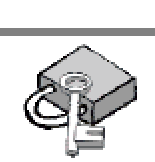
```
java.lang.Object
  └android.view.View
      └android.widget.ImageView
          └android.widget.ImageButton
```

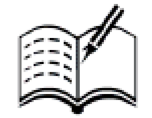
图 5-5　ImageButton 类继承关系

```
android:src="@drawable/picture"
```

也可以通过 setImageResource(int)方法来实现：

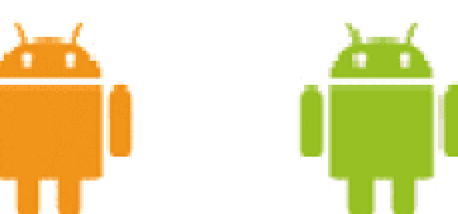
```
imageButton.setImageResource(R.drawable.picture);
```





Note

为了表示不同的按钮状态（焦点、选择等），可以为各种状态定义不同的图片。例如，如图 5-6 所示定义绿色图片（中间）为默认图片，橘黄色图片（左边）为获取焦点时显示的图片，黄色图片（右边）为按钮被按下时显示的图片，可以通过 XML 的 selector 配置来实现，代码如下：



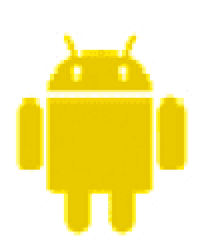


图 5-6　按钮不同状态所对应的图片

```
1 <?xml version="1.0" encoding="utf-8"?>
2       <selector xmlns:android="http://schemas.android.com/apk/res/android">
3             <item android:state  pressed="true"
4               android:drawable="@drawable/button  pressed" />
5             <item android:state  focused="true"
6               android:drawable="@drawable/button  focused" />
7             <item android:drawable="@drawable/button  normal" />
8  </selector>
```

说明：

- 第 2~8 行：声明一个 selector。
- 第 3~7 行：声明了 3 个 Item，分别用来表示当 ImageButton 按下、获得焦点以及默认情况下所显示的背景图片。第 3、4 行声明当 ImageButton 按下时显示 drawable 中名为 button_pressed 的图片；第 5、6 行声明当 ImageButton 获得焦点时显示 drawable 中名为 button_focused 的图片；第 7 行声明当 ImageButton 默认情况时显示 drawable 中名为 button_normal 的图片。

将上述代码复制保存到 drawable 文件夹下 bg.xml 文件中，以便后期使用。

5.2.3　按钮控件使用实例

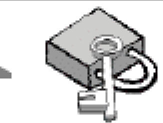
本节将通过实例来介绍按钮控件的使用方法。通过本实例，主要让读者了解 Button 如何设置监听器以及 ImageButton 如何使用 selector 来在不同的状态下显示不同的背景图片。

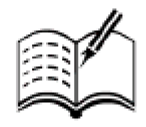
本实例开发步骤如下：

（1）创建项目 EX05_2。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button
```






```
9         android:text="按钮 1"
10        android:id="@+id/button1"
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13    />
14    <Button
15        android:text="按钮 2"
16        android:id="@+id/button2"
17        android:layout_width="wrap_content"
18        android:layout_height="wrap_content"
19        android:onClick="selfDestruct"
20    />
21    <ImageButton
22        android:id="@+id/imageButton1"
23        android:src="@drawable/bg"
24        android:layout_width="wrap_content"
25        android:layout_height="wrap_content"
26        android:background="#0000"
27    />
28    <EditText
29        android:layout_height="wrap_content"
30        android:layout_width="fill_parent"
31        android:id="@+id/editText1"
32    />
33    <EditText
34        android:layout_height="wrap_content"
35        android:layout_width="fill_parent"
36        android:id="@+id/editText2"
37    />
38    </LinearLayout>
```

说明：

- 第 8~20 行：声明了两个 Button。第 9、15 行分别设置了两个按钮显示的文本为“按钮 1”与“按钮 2”；第 10、16 行分别声明两个 Button 的 ID 为 button1 与 button2；第 11、17 行设置按钮的宽度为根据文本自适应；第 12、18 行设置按钮的高度为根据文本自适应；第 19 行声明了一个 selfDestruct 方法来替代 Activity 中的 OnClickListener。
- 第 21~27 行：声明了一个 ImageButton。第 22 行声明此 ImageButton 的 ID 为 imageButton1；第 23 行设置 ImageButton 的背景为 drawable 中名为 bg.xml 的文件，即前面我们复制并保存的 selector 配置文件；第 24、25 行分别设置 ImageButton 的宽度与高度都为根据内容自适应；第 26 行设置 ImageButton 的按钮背景为透明，在这里并不是背景图片的透明度，而是按钮的透明度。可以看到，若没有设置此属性，其效果如图 5-7 所示，而设置了此属性，其效果如图 5-8 所示。

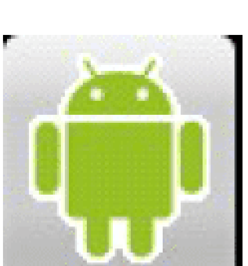
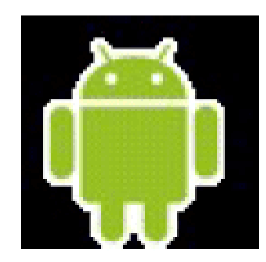


图 5-7　按钮背景不透明

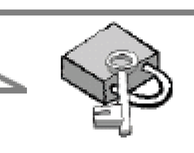

图 5-8　按钮背景透明

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，为 Button 控件设置监听器，并且为 ImageButton 设置 selector，以在不同的状态下显示不同的背景图片。编写代码如下：

```
1    package wyq.EX05_2;
2    import android.app.Activity;
3    import android.os.Bundle;
4    import android.view.View;
5    import android.view.View.OnClickListener;
6    import android.widget.Button;
7    import android.widget.EditText;
8    import android.widget.ImageButton;
9    public class FirstActivity extends Activity {
10           Button bt;
11           EditText et1,et2;
12           public void onCreate(Bundle savedInstanceState) {
13               super.onCreate(savedInstanceState);
14               setContentView(R.layout.main);
15               bt=(Button) findViewById(R.id.button1);
16               et1=(EditText) findViewById(R.id.editText1);
17               et2=(EditText) findViewById(R.id.editText2);
18               bt.setOnClickListener(new OnClickListener()
19           {
20                @Override
21                     public void onClick(View v)
22                     {
23                          et1.setText("消息来自 OnClickListener");
24                     }
25              });
26           }
27           public void selfDestruct(View view)
28           {
29            et2.setText("消息来自 selfDestruct");
30           }
31      }
```

说明：

- 第 10 行：声明了一个 Button 控件对象 bt。
- 第 11 行：分别声明了两个 EditText 控件对象 et1 与 et2。
- 第 15 行：通过 findViewById()获取 main.xml 布局文件中声明的 button1 控件。
- 第 16、17 行：通过 findViewById()获取 main.xml 布局文件中声明的 editText1 与



editText2。

- 第 18 行：为 Button 添加了一个 setOnClickListener 监听事件，并且在第 21 行设置了 onClick()方法，即在单击时触发此事件。
- 第 27~29 行：selfDestruct()方法对应 main.xml 布局文件中 button2 的 android:onClick 属性所声明的 selfDestruct 方法，即在单击 button2 时触发此事件。第 23、29 行设置在用户单击按钮后 EditText 所显示的文字，目的是为了让用户区分消息的来源。

本实例运行结果如图 5-9 所示。



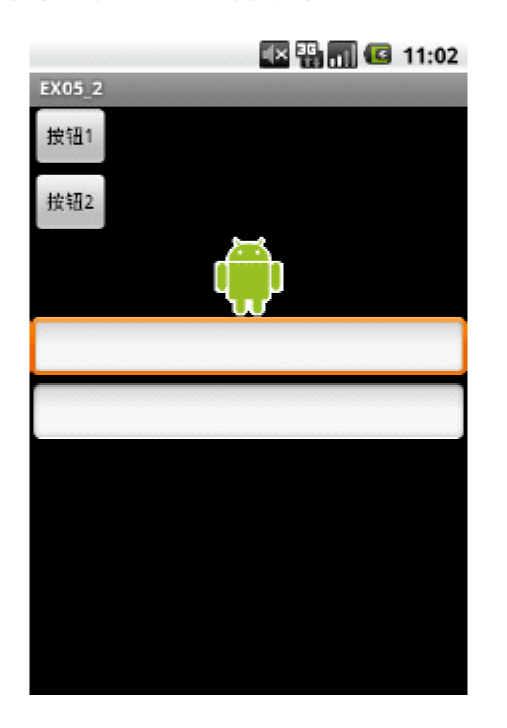




图 5-9　EX05_2 运行结果

5.3　单选按钮

在日常生活中经常会遇到二选一或者多选一的情况，例如，做一道单项选择题，这时就需要用到 Android 中提供的单选按钮。本节将对单选按钮的使用方法进行简单的介绍。

5.3.1　RadioButton 类简介

RadioButton 类的继承关系如图 5-10 所示。RadioButton 控件只有选中和未选中两种状态，并且在同一时刻一个 RadioGroup 中只能有一个按钮处于选中状态。

java.lang.Object
　└android.view.View
　　└android.widget.TextView
　　　└android.widget.Button
　　　　└android.widget.CompoundButton
　　　　　└android.widget.RadioButton

图 5-10　RadioButton 类继承关系

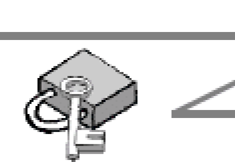
RadioButton 还包括一个常用的公共方法：

```
public void toggle()
```

其作用是将单选按钮更改为与当前选中状态相反的状态。如果该单选按钮已被选中，这个方法将不切换该单选按钮的状态。

5.3.2　单选按钮使用实例

本节将通过实例来介绍单选按钮的使用方法。在本实例中，利用 RadioButton 与





Note

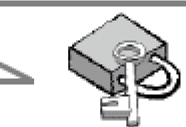
RadioGroup 模仿一道单项选择题界面，用户同一时刻可选且只能选择一个选项，在确定选项后可单击“确定”按钮来确认回答是否正确。

本实例开发步骤如下：

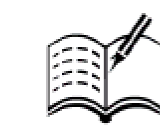
（1）新建项目 EX05_3。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
            android:layout_width=" fill _parent"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:text="下列说法中正确的是："
            />
             <RadioGroup
             android:id="@+id/radioGroup1"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content">
            <RadioButton
            android:layout_height="wrap_content"
            android:text="同一 RadioGroup 中同一时刻可选择多个 RadioButton"
            android:id="@+id/radio0"
            android:layout_width="wrap_content"
            android:checked="true"
              />
            <RadioButton
            android:layout_height="wrap_content"
            android:text="同一 RadioGroup 中同一时刻只能选择一个 RadioButton"
            android:id="@+id/radio1"
            android:layout_width="wrap_content"
            />
            <RadioButton
            android:layout_height="wrap_content"
            android:text="不同 RadioGroup 中同一时刻只能选择一个 RadioButton"
            android:id="@+id/radio2"
            android:layout_width="wrap_content"
            />
             </RadioGroup>
             <Button
             android:text="确定"
             android:id="@+id/button1"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             />
```



```
43          <TextView
44          android:layout_height="wrap_content"
45          android:id="@+id/TextView2"
46          android:text=""
47          android:layout_width="fill_parent"
48          />
49          </LinearLayout>
```

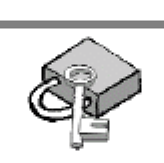


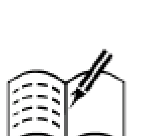
说明：

- 第 7~12 行：声明了一个宽度为填充父组件、高度为根据内容自适应且文字大小为 18sp 的 TextView 来显示题目。
- 第 13~36 行：声明了一个 RadioGroup，其高度和宽度都为根据其中内容自适应大小，其中包括 3 个 RadioButton，即 3 个选项。在此 RadioGroup 中的 RadioButton 在同一时刻能且只能有一个为选中状态。第 17~35 行分别声明了 3 个 RadioButton。
  - 第 18、25、31 行：分别定义 RadioButton 的高度为根据内容自适应。
  - 第 19、26、32 行：分别定义了 RadioButton 所显示的文字，在这里即需要显示的选项。
  - 第 20、27、33 行：分别定义 3 个 RadioButton 的 ID 为 radio0、radio1、radio2。
  - 第 21、28、34 行：分别定义 RadioButton 的宽度为根据内容自适应。
- 第 37~42 行：声明了一个宽度和高度都为自适应，显示文字为“确定”的按钮来允许用户提交确定的选项。
- 第 43~47 行：声明了一个宽度为填充父组件、高度为根据内容自适应的 TextView 来显示用户的选项是否正确。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，为 Button 控件设置监听器，判断单选按钮是否选中。编写代码如下：

```
package wyq.EX05_3;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.TextView;
public class FirstActivity extends Activity {
        /** Called when the activity is first created. */
        Button bt;
        RadioButton rb;
        TextView tv;
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            bt=(Button) findViewById(R.id.button1);
            rb=(RadioButton) findViewById(R.id.radio1);
```





Note

```
18              tv=(TextView) findViewById(R.id.TextView2);
19              bt.setOnClickListener(new OnClickListener()
20              {
21               @Override
22                   public void onClick(View v)
23                   {
                 // TODO Auto-generated method stub
24                       if(rb.isChecked())
25                           tv.setText("回答正确");
26                       else
27                           tv.setText("回答错误");
28                   }
29              });
30          }
31      }
```

说明：

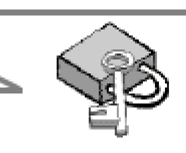
- ❑ 第 10 行：声明了一个 Button 控件对象 bt。后面用此控件来提交用户所选择的答案。
- ❑ 第 11 行：声明了一个 RadioButton 控件对象 rb。后面用此控件来判断是否选中了正确的选项。
- ❑ 第 12 行：声明了一个 TextView 控件对象 tv。后面用此控件来提示用户的选择是否正确。
- ❑ 第 16~18 行：分别通过 findViewById()获取 main.xml 布局文件中所声明的 button1、radio1 和 TextView2。
- ❑ 第 19 行：为 Button 添加了一个 setOnClickListener 监听事件，并且在第 22 行设置了 onClick()方法，即在单击时触发此事件。
- ❑ 第 24~27 行：按钮单击后的处理过程，即判断用户是否选择了正确的选项并且提示。

本实例运行结果如图 5-11 所示。

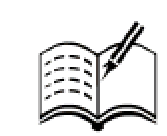




图 5-11　EX05_3 运行结果



5.4　复　选　框



在日常生活中也会遇到多选的情况，例如，用户选择兴趣爱好，这时单选按钮已经不能满足要求，就需要用到 Android 中提供的复选框。本节将对复选框的使用方法进行介绍。

5.4.1　CheckBox 类简介

CheckBox 类的继承关系如图 5-12 所示。CheckBox 控件也只有选中和未选中两种状态，但与 RadioButton 不同的是，CheckBox 同一时刻可以有多个按钮处于选中状态。

```
java.lang.Object
 └android.view.View
   └android.widget.TextView
     └android.widget.Button
       └android.widget.CompoundButton
         └android.widget.CheckBox
```

图 5-12　CheckBox 类继承关系

5.4.2　复选框使用实例

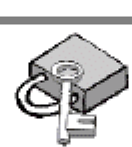
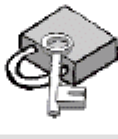
本节将通过一个实例来介绍复选框的使用方法。在本实例中，利用 CheckBox 模拟一个需要用户进行兴趣爱好选择的界面，兴趣爱好也许需要同时选择很多项，并且各项之间没有什么必然的联系，在此处利用 CheckBox 十分合适。

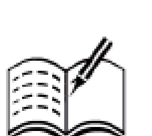
本项目的创建步骤如下：

（1）创建项目 EX05_4。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
        android:textSize="19sp"
        android:text="请选择您的兴趣爱好："
        />
        <CheckBox
        android:text="看书"
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
        <CheckBox
        android:text="听歌"
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
```


Note

```
24            <CheckBox
25            android:text="旅行"
26            android:id="@+id/checkBox3"
27            android:layout_width="wrap_content"
28            android:layout_height="wrap_content"
29            />
30            <CheckBox
31            android:text="游泳"
32            android:id="@+id/checkBox4"
33            android:layout_width="wrap_content"
34            android:layout_height="wrap_content"
35            />
36            <Button
37            android:text="确定"
38            android:id="@+id/bt_ok"
39            android:layout_width="wrap_content"
40            android:layout_height="wrap_content"
41            />
42            <TextView
43            android:layout_width="fill_parent"
44            android:layout_height="fill_parent"
45            android:id="@+id/tv_hobby"
46           />
47     </LinearLayout>
```

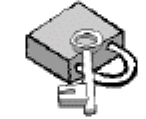
说明：

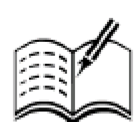
- 第 7~12 行：声明了一个 TextView 控件。在此处的作用是显示提示信息。第 8 行定义宽度为填充父组件；第 9 行高度随内容自适应；第 10 行设置 TextView 控件的定义字体大小为 19sp。
- 第 13~35 行：分别声明了 4 个 CheckBox 控件。
  - 第 14、19、25、31 行：分别定义了 CheckBox 的 android:text 属性所显示的文字，在这里是需要用户选择的兴趣爱好选项。
  - 第 15、20、26、32 行：分别定义了 4 个 CheckBox 的 ID 依次为 checkBox1、checkBox2、checkBox3、checkBox4。
  - 第 16、21、27、33 行：分别定义了 4 个 CheckBox 的宽度为根据文本自适应。
  - 第 17、22、28、34 行：分别定义了 4 个 CheckBox 的高度为根据文本自适应。
- 第 36~41 行：声明一个 ID 为 bt_ok 的 Button 控件。
- 第 42~46 行：声明一个 ID 为 tv_hobby 的 TextView 控件。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，显示所选择的兴趣爱好。编写代码如下：

```
package wyq.EX05_4;

import android.app.Activity;
import android.os.Bundle;
```

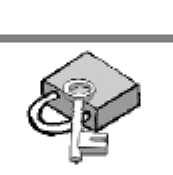





```
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.TextView;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private CheckBox cb1,cb2,cb3,cb4;
    private TextView tv_hobby;
    private Button bt_ok;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        cb1=(CheckBox)findViewById(R.id.checkBox1);
        cb2=(CheckBox)findViewById(R.id.checkBox2);
        cb3=(CheckBox)findViewById(R.id.checkBox3);
        cb4=(CheckBox)findViewById(R.id.checkBox4);
        tv_hobby=(TextView)findViewById(R.id.tv_hobby);
        bt_ok=(Button)findViewById(R.id.bt_ok);
        bt_ok.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String str_hobby="你的爱好有：\n";
                if(cb1.isChecked())
                {
                    str_hobby=str_hobby+cb1.getText().toString()+"\n";
                }
                if(cb2.isChecked())
                {
                    str_hobby=str_hobby+cb2.getText().toString()+"\n";
                }
                if(cb3.isChecked())
                {
                    str_hobby=str_hobby+cb3.getText().toString()+"\n";
                }
                if(cb4.isChecked())
                {
                    str_hobby=str_hobby+cb4.getText().toString();
                }
                tv_hobby.setText(str_hobby);
            }
        });
    }
}
```



说明：

- 第 12~14 行：定义 4 个 CheckBox 对象、1 个 TextView 对象与 1 个 Button 对象。
- 第 21~26 行：获取 CheckBox、TextView、Button 控件的引用。
- 第 27~51 行：为 bt_ok 按钮增加单击监听事件。根据选择的兴趣爱好，生成相应的字符串，并在 TextView 控件显示。

本实例运行结果如图 5-13 所示。





图 5-13　EX05_4 运行结果

5.5　图 片 控 件

本节将介绍图片控件 ImageView，首先对 ImageView 类进行简单的介绍，然后通过一个实例说明 ImageView 的用法。

5.5.1　ImageView 类简介

ImageView 类的继承关系如图 5-14 所示。ImageView 控件负责显示图片，其图片的来源既可以是资源文件的 ID，也可以是 Drawable 对象或 Bitmap 对象，还可以是 ContentProvider 的 Uri。

```
java.lang.Object
  └android.view.View
      └android.widget.ImageView
```

图 5-14　ImageView 类继承关系

ImageView 控件中常用到的属性及对应方法如表 5-3 所示，常用方法如表 5-4 所示。

表 5-3　ImageView 控件中常用属性及对应方法

属 性 名 称	对 应 方 法	说　明
android:adjustViewBounds	setAdjustViewBounds(boolean)	设置是否需要 ImageView 调整自己的边界来保证所显示的图片的长宽比例
android:maxHeight	setMaxHeight(int)	ImageView 的最大高度
android:maxWidth	setMaxWidth(int)	ImageView 的最大宽度
android:scaleType	setScaleType(ImageView.ScaleType)	控制图片应如何调整或移动来适合 ImageView 的尺寸
android:src	setImageResource(int)	设置 ImageView 要显示的图片



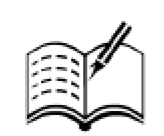


表 5-4　ImageView 控件常用方法

方法名称	说　明
setAlpha(int alpha)	设置 ImageView 的透明度
setImageBitmap(Bitmap bm)	设置 ImageView 所显示的内容为指定的 Bitmap 对象
setImageDrawable(Drawable drawable)	设置 ImageView 所显示的内容为指定的 Drawable 对象
setImageResource(int resId)	设置 ImageView 所显示的内容为指定 ID 的资源
setImageURI(Uri uri)	设置 ImageView 所显示的内容为指定的 Uri
setSelected(boolean selected)	设置 ImageView 的选中状态

5.5.2　图片控件使用实例

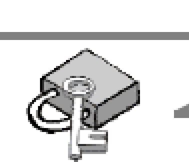
本节将通过一个实例来介绍图片控件的使用。在本实例中，使用 ImageView 显示一张图片，并且设置两个按钮，用户可以通过按钮来增加或者降低图片的透明度。

本实例的开发步骤如下：

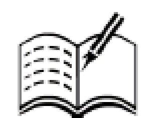
（1）创建项目 EX05_5。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <ImageView
            android:id="@+id/imageView1"
            android:layout_height="wrap_content"
            android:src="@drawable/pic"
            android:layout_width="wrap_content"
            android:layout_weight="0.9"
            />
            <LinearLayout
            android:layout_width=" fill_parent"
            android:layout_height="wrap_content">
                <Button
                android:text="透明度增加"
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:onClick="AlphaUp"
                />
                <Button
                android:text="透明度减少"
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```



```
29                  android:onClick="AlphaDown"
30                  />
31              </LinearLayout>
32          </LinearLayout>
```



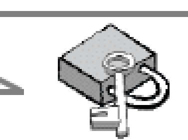
Note

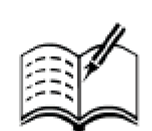
说明：

- 第 7~13 行：声明了一个 ImageView 控件。第 8 行定义此 ImageView 的 ID 为 imageView1；第 9 行定义此 ImageView 的高度根据其中的内容自适应大小；第 10 行定义此 ImageView 所要显示的图片来源为 drawable 下的 pic 文件；第 11 行定义宽度为填充父控件；第 12 行定义 ImageView 按原大小的 90%显示。
- 第 17~30 行：声明了两个 Button 控件，其大小都是根据内容自适应，并且通过 android:onClick 属性分别为两个 Button 添加了 AlphaUp 与 AlphaDown 两个方法。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，显示一张图片，并通过按钮来增加或者降低图片的透明度。编写代码如下：

```
package wyq.EX05  5;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
public class FirstActivity extends Activity {
/** Called when the activity is first created. */
        ImageView iv;
        int Alpha=255;
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            iv=(ImageView) findViewById(R.id.imageView1);
        }
        public void AlphaUp(View view)
        {
            if(Alpha<255)
            {
                Alpha=Alpha+5;
                iv.setAlpha(Alpha);
            }
        }
        public void AlphaDown(View view)
        {
            if(Alpha>0)
            {
                Alpha=Alpha-5;
                iv.setAlpha(Alpha);
            }
        }
```





Note

```
33    }
```

说明：

- 第 10 行：声明了一个 ImageView。
- 第 11 行：声明了一个整型变量 Alpha。其作用是记录 Alpha 的值，即透明度的值，Alpha 的取值范围为 0~255。
- 第 15 行：通过 findViewById()来获取 main.xml 布局文件中声明的 ImageView 控件。
- 第 17~24 行：AlphaUp()方法是单击 button1 时所触发的事件，其对应 main.xml 布局文件中 android:onClick="AlphaUp"所声明的方法。
- 第 25~32 行：AlphaDown()方法是单击 button2 时所触发的事件，其对应 main.xml 布局文件中 android:onClick="AlphaDown"所声明的方法。

本实例运行结果如图 5-15 所示。

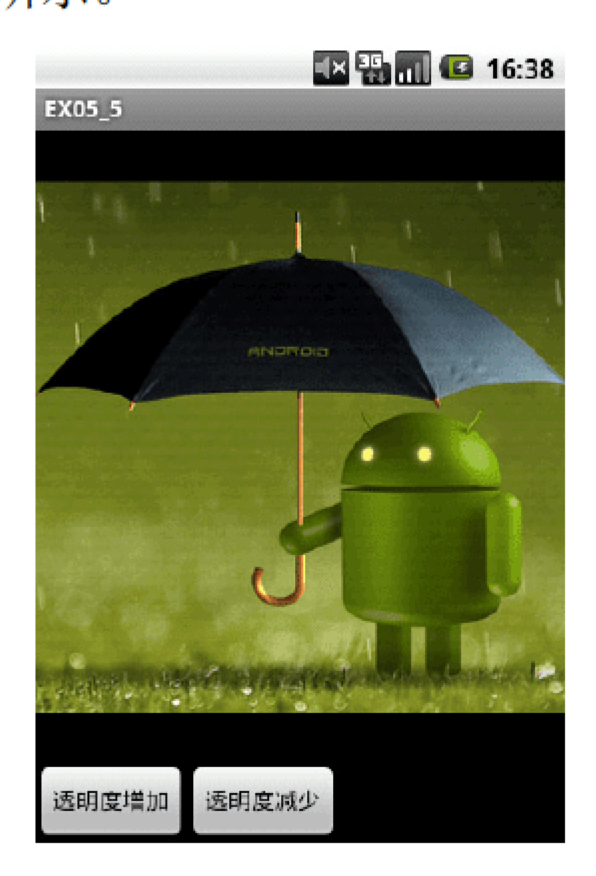


图 5-15　EX05_5 运行结果

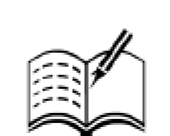
5.6　时钟控件

本节将对 Android 中的时钟控件进行介绍。时钟控件是 Android 用户界面中比较简单的控件，时钟控件包括 AnalogClock 和 DigitalClock 控件。下面先介绍 AnalogClock 类和 DigitalClock 类，然后通过实例来说明时钟控件的用法。

5.6.1　AnalogClock 类与 DigitalClock 类简介

AnalogClock 类继承于 android.View 类，是一个能够显示时与分的模拟时钟。

DigitalClock 类继承于 widget.TextView 类。DigitalClock 与 AnalogClock 不同，DigitalClock 是一个数字时钟，能够精确到秒，但却不能像 AnalogClock 一样模拟真实的钟表转动效果。



Note

5.6.2 时钟控件使用实例

本节将通过一个实例来演示时钟控件的使用方法。在本实例中，界面上分别放置了一个 AnalogClock 控件与一个 DigitalClock 控件，让读者更加直观地理解 AnalogClock 控件与 DigitalClock 控件的区别。

本实例的开发步骤如下：

（1）新建项目 EX05_6。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       >
7           <DigitalClock
8           android:layout_width="fill_parent"
9           android:layout_height="wrap_content"
10          android:textSize="19sp"
11          android:id="@+id/analogClock2"
12          />
13          <AnalogClock
14          android:layout_width="fill_parent"
15          android:layout_height="wrap_content"
16          android:id="@+id/analogClock1"
17          />
18      </LinearLayout>
```

说明：

- 第 7~12 行：声明了一个 DigitalClock。第 8 行定义宽度为填充父控件；第 9 行定义高度为随内容自适应；第 10 行设置 DigitalClock 控件的字体大小为 19sp。
- 第 13~17 行：声明了一个 AnalogClock 控件。第 14 行定义宽度为填充父控件；第 15 行定义高度为随内容自适应。

本实例运行结果如图 5-16 所示。



图 5-16　EX05_6 运行结果



5.7　日期与时间控件

本节介绍日期与时间控件，首先对 DatePicker 和 TimePicker 类进行介绍，然后通过实例来说明如何在程序中使用日期与时间控件。

5.7.1　DatePicker 类简介

DatePicker 类的继承关系如图 5-17 所示。DatePicker 控件的主要功能是向用户提供包含年、月、日的日期数据并允许用户对其进行选择。如果要捕获用户修改的日期选择控件中数据的事件，需要为 DatePicker 添加 onDateChangedListener 监听器。其常用方法如表 5-5 所示。

```
java.lang.Object
  └android.view.View
     └android.view.ViewGroup
        └android.widget.FrameLayout
           └android.widget.DatePicker
```

图 5-17　DatePicker 类继承关系

表 5-5　DatePicker 常用方法

方 法 名 称	说　明
getDayOfMonth()	获取日期天数
getMonth()	获取日期月份
getYear()	获取日期年份
setEnabled(boolean enabled)	设置控件是否可用
updateDate(int year, int month, int dayOfMonth)	根据传入的参数更新日期选择控件的各个属性值

5.7.2　TimePicker 类简介

TimePicker 类的继承关系如图 5-18 所示。TimePicker 控件向用户显示一天中的时间，并允许用户进行选择，如果要捕获用户修改时间数据的事件，需要为 TimePicker 添加 OnTimeChangedListener 监听器。其常用方法如表 5-6 所示。

```
java.lang.Object
  └android.view.View
     └android.view.ViewGroup
        └android.widget.FrameLayout
           └android.widget.TimePicker
```

图 5-18　TimePicker 类继承关系

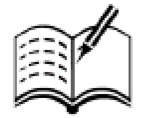
表 5-6　TimePicker 常用方法

方 法 名 称	说　明
getCurrentHour()	获取时间选择控件的当前小时
getCurrentMinute()	获取时间选择控件的当前分钟
is24HourView()	判断控件是否为 24 小时制
setCurrentHour(Integer currentHour)	设置时间选择控件的当前小时
setCurrentMinute(Integer currentMinute)	设置时间选择控件的当前分钟
setEnabled(boolean enabled)	设置控件是否可用



续表

方 法 名 称	说　　明
setIs24HourView(Boolean is24HourView)	设置控件是否为 24 小时制
setOnTimeChangedListener (TimePicker.OnTimeChangedListener)	为时间选择控件添加 OnTimeChangedListener 监听器



Note

5.7.3　日期与时间控件使用实例

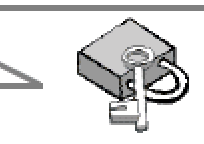
本节将通过一个实例来介绍日期与时间控件的使用方法。本实例中，在界面上分别放置一个 DatePicker 控件与一个 TimePicker 控件，然后通过两个 Button 来获取设置的日期与时间。

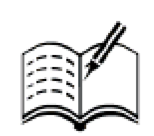
本实例的开发步骤如下：

（1）新建项目 EX05_7。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
     >
<DatePicker
     android:id="@+id/datePicker1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
    />
    <Button
        android:text="获取日期"
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="getDate"
    />
    <TimePicker
        android:id="@+id/timePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    />
    <Button
        android:text="获取时间"
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="getTime"
    />
    </LinearLayout>
```





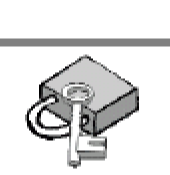
Note

说明：

- 第 7~11 行：声明了一个 DatePicker 控件。第 8 行定义此 DatePicker 的 ID 为 datePicker1；第 9 行定义宽度为随内容自适应；第 10 行定义高度为随内容自适应。
- 第 12~18 行：声明了一个 Button 控件，其宽度和高度都根据内容自适应，并且通过 android:onClick 属性添加了 getDate()方法。
- 第 19~23 行：声明了一个 TimePicker 控件。第 20 行定义此 TimePicker 的 ID 为 timePicker1；第 21 行定义宽度为随内容自适应；第 22 行定义高度为随内容自适应。
- 第 24~30 行：声明了一个 Button 控件，其宽度和高度都根据内容自适应，并且通过 android:onClick 属性添加了 getTime()方法。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，显示 DatePicker 与 TimePicker 控件，然后获取设置的日期与时间。编写代码如下：

```
package wyq.EX05_7;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TimePicker;
public class FirstActivity extends Activity {
 /** Called when the activity is first created. */
        DatePicker dp;
        TimePicker tp;
        Button getdate,gettime;
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            dp=(DatePicker) findViewById(R.id.datePicker1);
            tp=(TimePicker) findViewById(R.id.timePicker1);
            getdate=(Button) findViewById(R.id.button1);
            gettime=(Button) findViewById(R.id.button2);
        }
        public void getDate(View v)
        {
         String date;
         date=dp.getYear()+"年"+dp.getMonth()+"月"+dp.getDayOfMonth();
         getdate.setText(date);
        }
    public void getTime(View v)
        {
         String time;
         time=tp.getCurrentHour()+":"+tp.getCurrentMinute();
         gettime.setText(time);
        }
    }
```



Note

说明：

- 第 9 行：声明了一个 DatePicker 控件 dp。
- 第 10 行：声明了一个 TimePicker 控件 tp。
- 第 11 行：声明了两个 Button 控件对象 getdate 与 gettime。
- 第 15 行：通过 findViewById()获取 main.xml 布局文件中声明的 datePicker1。
- 第 16 行：通过 findViewById()获取 main.xml 布局文件中声明的 timePicker1。
- 第 17 行：通过 findViewById()获取 main.xml 布局文件中声明的 button1。
- 第 18 行：通过 findViewById()获取 main.xml 布局文件中声明的 button2。
- 第 20~25 行：为 Activity 添加 getDate()方法来对应 main.xml 布局文件中 android:onClick="getDate"所声明的方法。第 22 行声明一个名为 date 的字符串变量来存储获取的日期；第 23 行通过 getYear()、getMonth()和 getDayOfMonth()方法来获取用户在 DatePicker 中所设置的日期并且存储在字符串变量 date 中。第 24 行通过 setText(text)方法来设置 Button 的 Text 以便显示用户设置的日期。
- 第 26~32 行：为 Activity 添加 getTime()方法来对应 main.xml 布局文件中 android:onClick="getTime"所声明的方法。第 28 行声明一个名为 time 的字符串变量来存储获取的时间；第 29 行通过 getCurrentHour()和 getCurrentMinute()方法来获取用户在 TimePicker 中所设置的时间并且存储在字符串变量 time 中；第 30 行通过 setText(text)方法来设置 Button 的 Text 以便显示用户设置的时间。

本实例运行结果如图 5-19 所示。



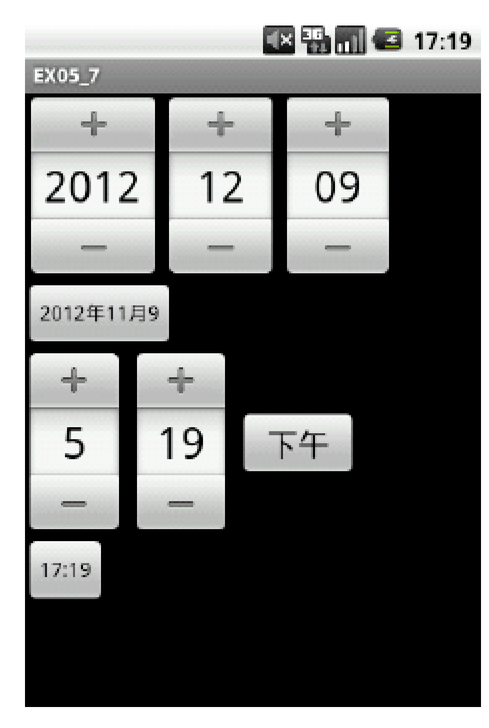


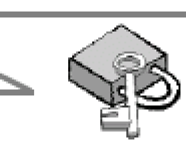
图 5-19　EX05_7 运行结果

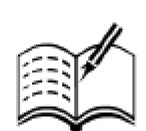
5.8　习　　题

1．在 Android 应用程序中实现如下功能：在 TextView 控件中显示通过 EditText 控件输入的内容；通过 Button 命令按钮，更改 TextView 控件的文字及大小。

2．在 Android 应用程序中，设计具有背景图片的按钮，并且根据按钮的状态显示不同的背景图片。

3．设计一个图书选购程序，在该程序中，选中图书后，单击“确定”按钮，在屏幕



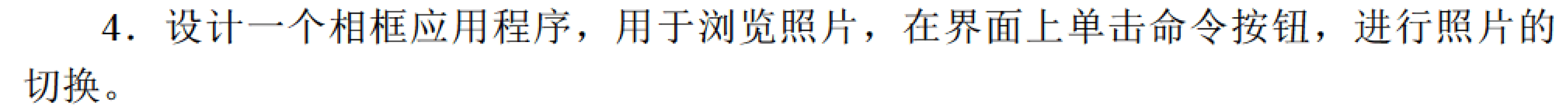


Note

的下方显示所选择的图书，界面如图 5-20 所示。




图 5-20　图书选购程序


4．设计一个相框应用程序，用于浏览照片，在界面上单击命令按钮，进行照片的切换。

5．设计一个 Android 程序，实现以下功能：（1）在界面上显示数字和模拟时钟，默认显示手机的当前系统时间；（2）通过日期、时间控件设置时间，并且在数字和模拟时钟中显示。

第 6 章 高级控件

【本章内容】

- 自动完成文本框
- 下拉列表控件
- 滚动视图
- 列表视图
- 网格视图
- 滑块与进度条
- 选项卡
- 画廊控件

在第 5 章介绍了 Android 一些常用的基本控件，除了这些常用的控件之外，Android 还提供了一些功能更强大的控件。本章将通过实例对自动完成文本框、下拉列表控件、滚动视图、列表视图、网格视图、滑块与进度条、选项卡、画廊控件等高级控件进行介绍。

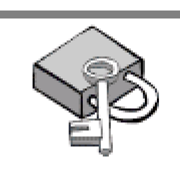
6.1 自动完成文本框

在使用网络搜索引擎输入关键字时，只要输入几个文字，就会显示一些相关的关键字供用户选择。通过这一功能，可以减少用户的输入，提高用户体验。在 Android 中，这一功能可以通过自动完成文本框很轻松地完成。自动完成文本框有两种：AutoCompleteTextView 与 MultiAutoCompleteTextView，两者之间的区别为，AutoCompleteTextView 每次只能选择一个选项，而 MultiAutoCompleteTextView 可以选择多个选项。下面进行详细介绍。

6.1.1 AutoCompleteTextView 类简介

自动完成文本框是一个当用户输入时显示自动完成建议的可编辑文本视图。自动完成建议显示在一个下拉列表中，用户可以选择一个项目，以取代在编辑框中的内容，也可以按 Esc 或者 Backspace 键取消下拉列表。

AutoCompleteTextView 类继承于 android.widget.EditText 类。下面对该类的常用属性及对应方法进行介绍，如表 6-1 所示。



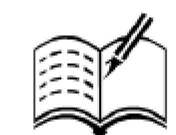


表 6-1 AutoCompleteTextView 常用属性及对应方法

属　　性	对 应 方 法	说　　明
android:completionThreshold	setThreshold(int)	设置显示自动提示需要输入的字符数
android:dropDownHeight	setDropDownHeight(int)	设置下拉列表的高度，建议使用默认值
android:dropDownWidth	setDropDownWidth(int)	设置下拉列表的宽度，建议使用默认值
android:popupBackground	setDropDownBackgroundResource(int)	设置下拉列表的背景

如果要使用自动完成文本框控件，需要通过以下步骤：

（1）定义一个字符串数组。

（2）将此字符串数组放入数组适配器（ArrayAdapter）。

（3）利用 AutoCompleteTextView 的 setAdapter()方法，将字符串数组加入到 AutoCompleteTextView 对象中，设置自动完成文本框的适配器。

6.1.2 MultiAutoCompleteTextView 类简介

MultiAutoCompleteTextView 类继承于 AutoCompleteTextView 类，所以其属性、方法与 AutoCompleteTextView 类似，这里不再进行介绍。MultiAutoCompleteTextView 允许一次选择多个选项，所以在编程方法上与 AutoCompleteTextView 稍有不同，在设置完控件的适配器之后，必须提供一个 MultiAutoCompleteTextView.Tokenizer 来区分不同的子串。

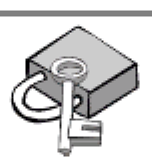
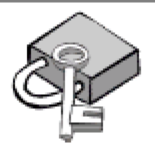
6.1.3 自动完成文本框使用实例

本节将通过实例来演示自动完成文本框的使用方法。本实例的开发步骤如下：

（1）创建项目 EX06_1。

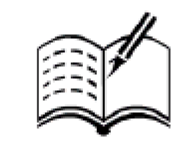
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="这是一个自动完成文本框实例："
    />
<AutoCompleteTextView
     android:id="@+id/myAutoCompleteTextView"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:hint="请输入您需要的城市名称"
     android:completionHint="我知道的城市"
     />
<MultiAutoCompleteTextView
```



```
20         android:id="@+id/myMulti"
21         android:layout_width="fill_parent"
22         android:layout_height="wrap_content"
23         />
24 </LinearLayout>
```




说明：

- ❑ 第 2~6 行：对线性布局进行设置，布局方向为垂直方向，宽度和高度自适应父控件，即窗口。
- ❑ 第 7~11 行：在线性布局中添加一个 TextView 控件。
- ❑ 第 12~18 行：在线性布局中添加一个单选自动完成文本框，其 ID 为 myAutoCompleteTextView，第 16 行代码设置在没有控件、没有输入任何内容时的提示文本。
- ❑ 第 19~23 行：在线性布局中添加一个多选自动完成文本框，其 ID 为 myMulti。

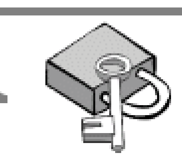
（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

```
1 package wyq.EX06_1;
2 import android.app.Activity;
3 import android.os.Bundle;
4 import android.widget.ArrayAdapter;
5 import android.widget.AutoCompleteTextView;
6 import android.widget.MultiAutoCompleteTextView;
7 public class FirstActivity extends Activity {
8      /** Called when the activity is first created. */
9     private String[] autoStr={"beijing","shanghai","shenzhen","xi'an"};
10     private AutoCompleteTextView myAutoTextView;
11     private MultiAutoCompleteTextView myMultiTextView;
12     @Override
13     public void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.main);
16        ArrayAdapter<String> ada=new ArrayAdapter<String>(this, android.R.layout. simple_ dropdown
          _item_1line,autoStr);
17         myAutoTextView=(AutoCompleteTextView)findViewById(R.id.myAutoComplete TextView);
18         myAutoTextView.setAdapter(ada);
19         myAutoTextView.setThreshold(1);
20         myMultiTextView=(MultiAutoCompleteTextView)findViewById(R.id.myMulti);
21         myMultiTextView.setAdapter(ada);
22          myMultiTextView.setTokenizer(new MultiAutoCompleteTextView. CommaTokenizer());
23          myMultiTextView.setThreshold(1);
24     }
25 }
```

说明：

- ❑ 第 2~6 行：说明本实例引入的类。
- ❑ 第 9 行：定义自动完成文本框显示项目的数组，作为适配器的资源数组。







- 第 16 行：创建数组适配器。在创建适配器时，使用的是 Android 系统自带的简单布局 android.R.layout.simple_dropdown_item_1line，然后将第 9 行定义的资源数组作为适配器的数据源。
- 第 17~19 行：先得到单选自动完成文本框的引用，然后设置其适配器为第 16 行所创建的适配器，并设置显示自动提示需要输入的字符数。
- 第 20~23 行：先得到多选自动完成文本框的引用，然后设置其适配器为第 16 行所创建的适配器，并设置显示自动提示需要输入的字符数。

本实例运行后，在自动完成文本框中输入字母“s”，结果如图 6-1 和图 6-2 所示。

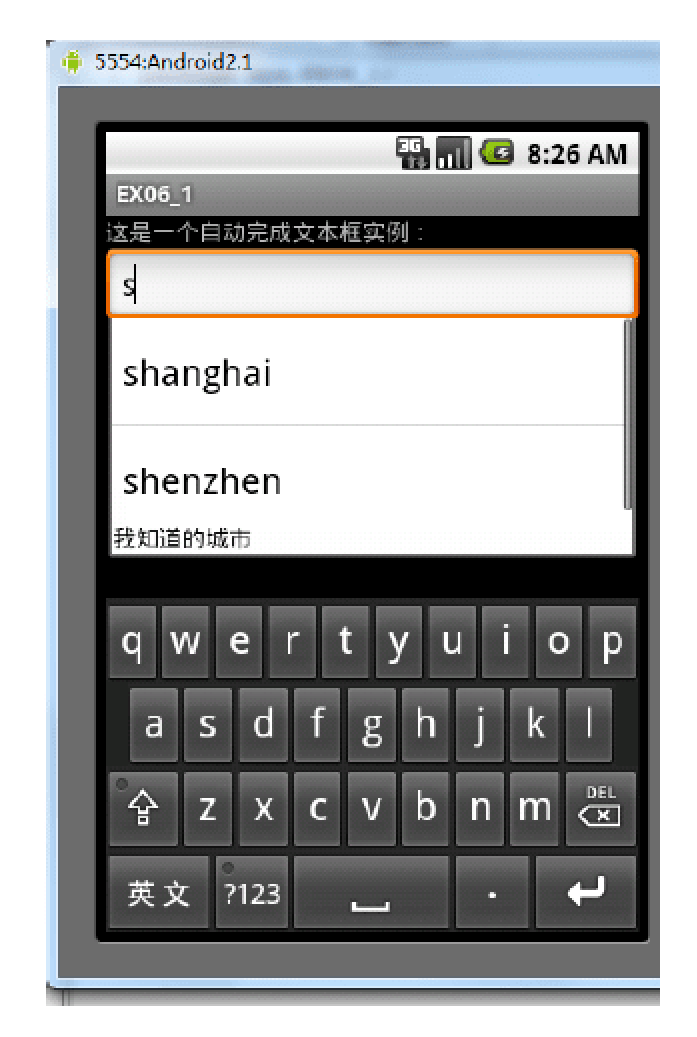


图 6-1　单选自动完成文本框



图 6-2　多选自动完成文本框

6.2　下拉列表控件

下拉列表是 Android 应用程序开发最常用的控件之一，用来从多个选项中选择一项，例如城市的选择等。下面进行详细介绍。

6.2.1　Spinner 类简介

Spinner 类位于 android.widget 包下。当用户单击该控件时，弹出选择列表供用户选择，并且只能选择其中一项，选择列表中的选项来自于该 Spinner 控件的适配器。Spinner 类的属性比较简单，如表 6-2 所示。

表 6-2　Spinner 类属性

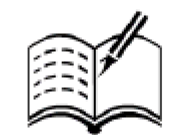
属　性	说　明
android:prompt	设置下拉列表对话框显示时的提示

如果要使用下拉列表控件，需要通过以下步骤：

（1）定义一个字符串数组。

（2）将此字符串数组放入数组适配器（ArrayAdapter）。

（3）利用 Spinner 的 setAdapter()方法，将适配器加入到 Spinner 对象中，设置下拉列表控件的适配器。


Note

6.2.2　下拉列表控件使用实例

本节将通过实例来演示下拉列表控件的使用方法。在本实例中，从下拉列表中选择一个城市，然后显示所选择的城市。本实例的开发步骤如下：

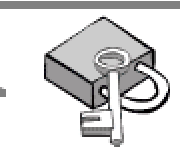
（1）创建项目 EX06_2_01。

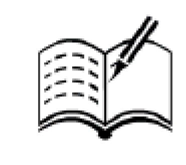
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="这是一个 Spinner 实例"
11     android:textSize="20px"
12     />
13 <TextView
14     android:id="@+id/tv"
15     android:layout_width="fill_parent"
16     android:layout_height="wrap_content"
17     android:text="请选择城市:"
18     android:textSize="20px"
19     />
20 <Spinner
21     android:id="@+id/citySpinner"
22     android:layout_width="fill_parent"
23     android:layout_height="wrap_content"
24     />
25 <TextView
26     android:id="@+id/cityResult"
27     android:layout_width="fill_parent"
28     android:layout_height="wrap_content"
29     android:textSize="20px"
30     />
31 </LinearLayout>
```

说明：

- 第 2~6 行：对线性布局进行设置，布局方向为垂直方向，宽度和高度自适应父控件，即窗口。
- 第 7~12 行：在线性布局中添加一个 TextView 控件。





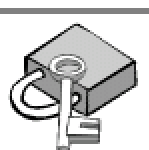
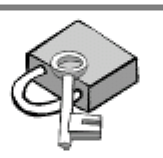
- ❑ 第 13~19 行：在线性布局中添加一个 TextView 控件，并设置其 ID 为 tv。
- ❑ 第 20~24 行：在线性布局中添加一个 Spinner 控件，其 ID 为 citySpinner。
- ❑ 第 25~30 行：在线性布局中添加一个 TextView 控件，其 ID 为 cityResult，并且设置该控件的字体大小为 20px。

（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

```
1 package wyq.EX06_2_01;
2 import android.app.Activity;
3 import android.os.Bundle;
4 import android.view.View;
5 import android.widget.AdapterView;
6 import android.widget.ArrayAdapter;
7 import android.widget.Spinner;
8 import android.widget.TextView;
9 public class FirstActivity extends Activity {
10     /** Called when the activity is first created. */
11     private TextView tv;
12     private Spinner citySpinner;
13     private String [] cityList={"北京","上海","天津","重庆","西安"};
14     @Override
15     public void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.main);
18         tv=(TextView)findViewById(R.id.cityResult);
19         citySpinner=(Spinner)findViewById(R.id.citySpinner);
20         ArrayAdapter<String> spinnerAda=new ArrayAdapter<String>(this,
               android.R.layout.simple_spinner_item, cityList);
21         citySpinner.setAdapter(spinnerAda);
22         citySpinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener()
23         {
24             @Override
25          public void onItemSelected(AdapterView<?> arg0, View arg1,int arg2, long arg3) {
26                tv.setText("你选择的城市是:"+cityList[arg2]);
27             }
28             @Override
29         public void onNothingSelected(AdapterView<?> arg0) {}
30         });
31     }
32 }
```

说明：

- ❑ 第 2~8 行：说明本实例引入的类。
- ❑ 第 13 行：定义 Spinner 要显示项目的数组，作为适配器的资源数组。
- ❑ 第 18 行：获取 TextView 控件的引用。
- ❑ 第 19 行：获取 Spinner 控件的引用。
- ❑ 第 20 行：创建数组适配器。在创建适配器时，使用的是 Android 系统自带的简单

布局 android.R.layout.simple_spinner_item，然后将第 13 行定义的资源数组传入。

- 第 21 行：将 Spinner 控件适配器设置为第 20 行所创建的适配器。
- 第 22~31 行：为 Spinner 控件添加 setOnItemSelectedListener 监听事件。其中第 25~28 行重写 onItemSelected()函数，在 TextView 中显示所选择的城市；第 29 行重写 onNothingSelected()函数，虽然该函数是一个空函数，但是不能省略。

本实例运行结果如图 6-3 所示，单击向下箭头后结果如图 6-4 所示。



图 6-3　EX06_2_01 运行结果



图 6-4　单击向下箭头后的结果

6.2.3　动态添加/删除 Spinner 列表

在 6.2.2 节的实例中，对自定义 Spinner 下拉菜单及其交互事件进行了介绍，但是 Spinner 中的选择项不能动态添加、删除。本节将介绍如何动态添加、删除 Spinner 的列表项。在本节的实例中，设计一个 EditText，用户输入新的城市，然后单击“添加”按钮，将所输入的内容添加到 Spinner 的列表中，同时在 TextView 中显示添加的选项；当单击“删除”按钮时，则删除所选择的 Spinner 选项。

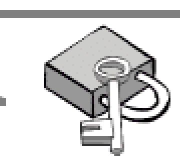
本实例开发步骤如下：

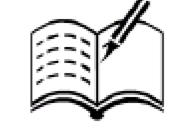
（1）创建项目 EX06_2_02。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="这是一个 Spinner 实例"
     android:textSize="20px"
     />
<TextView
     android:id="@+id/tv"
```





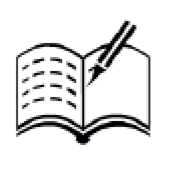


Note

```
15     android:layout_width="fill_parent"
16     android:layout_height="wrap_content"
17     android:text="请选择城市:"
18     android:textSize="20px"
19     />
20 <Spinner
21     android:id="@+id/citySpinner"
22     android:layout_width="fill_parent"
23     android:layout_height="wrap_content"
24     />
25 <TextView
26     android:id="@+id/cityResult"
27     android:layout_width="fill_parent"
28     android:layout_height="wrap_content"
29     android:textSize="20px"
30     />
31 <EditText
32     android:id="@+id/newCity"
33     android:layout_width="fill_parent"
34     android:layout_height="wrap_content"
35     />
36 <Button
37     android:id="@+id/btAddCity"
38     android:layout_width="fill_parent"
39     android:layout_height="wrap_content"
40     android:text="增加"
41     />
42 <Button
43     android:id="@+id/btDelCity"
44     android:layout_width="fill_parent"
45     android:layout_height="wrap_content"
46     android:text="删除"
47     />
48 </LinearLayout>
```

说明：

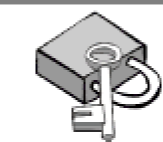
- 第 2~6 行：对线性布局进行设置，布局方向为垂直方向，宽度和高度自适应父控件，即窗口。
- 第 7~12 行：在线性布局中添加一个 TextView 控件。
- 第 13~19 行：在线性布局中添加一个 TextView 控件，并设置其 ID 为 tv。
- 第 20~24 行：在线性布局中添加一个 Spinner 控件，其 ID 为 citySpinner。
- 第 25~30 行：在线性布局中添加一个 TextView 控件，其 ID 为 cityResult，并且设置该控件的字体大小为 20px。
- 第 31~35 行：在线性布局中添加一个 EditText 控件，其 ID 为 newCity，用于输入新的城市名称。
- 第 36~41 行：在线性布局中添加一个 Button 控件，其 ID 为 btAddCity。

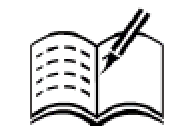

Note

- 第 42~47 行：在线性布局中添加一个 Button 控件，其 ID 为 btDelCity。

（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

```
package wyq.EX06_2_02;
import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private TextView tv;
    private Spinner citySpinner;
    private String [] cityList={"北京","上海","天津","重庆","西安"};
    private EditText newCity;
    private Button btAdd;
    private Button btDel;
    private ArrayList<String> allCityList;
    private ArrayAdapter<String> spinnerAda;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tv=(TextView)findViewById(R.id.cityResult);
        citySpinner=(Spinner)findViewById(R.id.citySpinner);
        newCity=(EditText)findViewById(R.id.newCity);
        btAdd=(Button)findViewById(R.id.btAddCity);
        btDel=(Button)findViewById(R.id.btDelCity);
        allCityList=new ArrayList<String>();
        for(String city :cityList)
        {
            allCityList.add(city);
        }
        spinnerAda=new ArrayAdapter<String>(this, android.R.layout.
                        simple_spinner_item, allCityList);
        citySpinner.setAdapter(spinerAda);
        citySpinner.setPrompt("请选择城市：");
        citySpinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener()
        {
         @Override
         public void onItemSelected(AdapterView<?> arg0, View arg1,int arg2, long arg3) {
             tv.setText("你选择的城市是:"+arg0.getSelectedItem().toString());
        }
         @Override
```

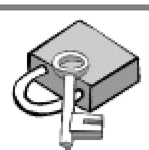
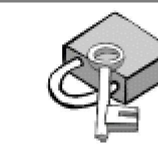





```
46          public void onNothingSelected(AdapterView<?> arg0) {}
47        });
48        btAdd.setOnClickListener(new Button.OnClickListener()
49        {
50         @Override
51         public void onClick(View v) {
52             String txtnewCity=newCity.getText().toString();
53             for(String city :cityList)
54             {
55                 if(txtnewCity==city)
56                     return;
57             }
58             if(txtnewCity!="")
59             {
60                 spinnerAda.add(txtnewCity);
61                 int pos=spinnerAda.getPosition(txtnewCity);
62                 citySpinner.setSelection(pos);
63                 newCity.setText("");
64             }
65         }
66        });
67        btDel.setOnClickListener(new Button.OnClickListener()
68        {
69         @Override
70         public void onClick(View v) {
71               spinnerAda.remove(citySpinner.getSelectedItem().toString());
72               tv.setText("你选择的城市是:");
73               newCity.setText("");
74         }
75        });
76     }
77 }
```

说明：

- 第 2~11 行：说明本实例引入的类。
- 第 16 行：定义 Spinner 要显示项目的数组。
- 第 26~30 行：使用 findViewById()获取 TextView、Spinner、EditText、Button 等控件的引用。
- 第 31~35 行：定义一个字符串列表，第 32~35 行将第 16 行定义的数组元素放入该 List 中。因为本实例要动态增加、删除 Spinner 的项目，使用字符数组不能满足要求，所以需要将数组中的元素放入 List 中，进行动态的增加、删除。
- 第 36 行：创建数组适配器。在创建适配器时，使用的是 Android 系统自带的简单布局 android.R.layout.simple_spinner_item，然后将第 31 行定义的列表作为适配器的数据源。
- 第 37 行：将 Spinner 控件适配器设置为第 36 行所创建的适配器。
- 第 38 行：设置当 Spinner 对话框关闭时显示的提示。

Note

- 第 39~47 行：为 Spinner 控件添加 setOnItemSelectedListener 监听事件。其中第 42~44 行重写 onItemSelected()函数，在 TextView 中显示所选择的城市；第 46 行重写 onNothingSelected()函数，虽然该函数是一个空函数，但是不能省略。
- 第 48~66 行：为命令按钮控件 btAdd 增加单击监听事件 setOnClickListener，用于将新输入的城市名称增加到 Spinner 中。在增加新的城市名称前，先检查所输入的城市名称在 Spinner 的项目中是否存在，如果不存在，则添加到 Spinner 的项目中，否则不添加。第 50~64 行重写 onClick()函数；第 52 行获取新输入的城市名称；第 53~57 行检查所输入的城市名称在 Spinner 的项目中是否存在，如果存在则返回；第 58~64 行将新输入的城市名称增加到 Spinner 中，并在 Spinner 中显示新增加的城市。
- 第 67~76 行：为命令按钮控件 btDel 增加单击监听事件 setOnClickListener，用于删除 Spinner 中的项目。第 71 行从 Spinner 中移除所选择的 Item。

本实例运行结果如图 6-5 所示，单击“增加”按钮添加城市结果如图 6-6 所示，单击“删除”按钮删除城市结果如图 6-7 所示。



图 6-5　EX06_2_02 运行结果



图 6-6　添加城市



图 6-7　删除城市

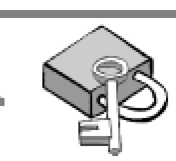
6.3　滚动视图

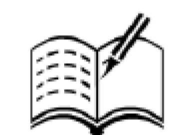
当应用程序的界面上控件比较多时，手机屏幕可能显示不下。此时，可以使用滚动视图 ScrollView 来滚动显示屏幕的控件。下面进行详细介绍。

6.3.1　ScrollView 类介绍

滚动视图是一种可供用户滚动的层次结构布局容器，允许显示比实际多的内容。ScrollView 类继承自 FrameLayout，所以需要在其上放置有滚动内容的子元素。子元素可以是一个复杂的对象的布局管理器，通常使用垂直方向的 LinearLayout。

TextView 类也有自己的滚动功能，所以不需要使用 ScrollView，但只有两者结合使用才可以实现在一个较大的容器中的文本视图效果。





Note

6.3.2　滚动视图使用实例

滚动视图可以在布局文件中进行配置，也可以通过 Java 代码进行设置。本节的实例将通过在布局文件中进行配置实现。本实例开发步骤如下：

（1）创建项目 EX06_3。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!-- 定义 ScrollView，为里面的组件添加垂直滚动条 -->
3 <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent" >
6     <!-- 定义 HorizontalScrollView，为里面的组件添加水平滚动条 -->
7     <HorizontalScrollView
8       android:layout_width="fill_parent"
9       android:layout_height="wrap_content" >
10        <LinearLayout
11           android:layout_width="wrap_content"
12           android:layout_height="fill_parent"
13           android:orientation="vertical" >
14           <TextView
15              android:layout_width="wrap_content"
16              android:layout_height="wrap_content"
17              android:text="这是一个滚动视图的实例"
18              android:textSize="40dp" />
       …
67        </LinearLayout>
68     </HorizontalScrollView>
69 </ScrollView>
```

说明：

- 第 3 行：在主 Activity 中定义一个滚动视图，并定义滚动视图的大小。
- 第 7~9 行：在滚动视图中定义一个水平滚动条，并定义其大小。
- 第 10~13 行：在滚动视图中定义一个纵向的线性布局，并定义其大小。
- 第 14~18 行：在线性布局中定义一个 TextView 控件，并定义其大小、文字及字体大小。

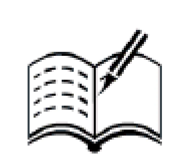
本实例运行结果如图 6-8 和图 6-9 所示。



图 6-8　Ex06_3 运行结果



图 6-9　屏幕向右滚动


Note

6.4 列表视图

列表视图 ListView 是 Android 应用程序开发中常用的一个控件，它可以根据屏幕的大小，把具体的内容以列表的形式显示出来，如电话本、通话记录等。本节将对列表视图进行介绍，并通过实例来演示列表视图的使用方法。

6.4.1 ListView 类简介

ListView 类位于 android.widget 包下，是一种列表视图，用于将适配器所提供的内容显示在一个垂直且可滚动的列表中。ListView 的常用属性及对应方法如表 6-3 所示。

表 6-3 ListView 常用属性及对应方法

属　性	相 关 方 法	说　明
android:choiceMode	setChoiceMode (int choiceMode)	规定此 ListView 所使用的选择模式。默认状态下，list 没有选择模式。属性值必须设置为下列常量之一：none，值为 0，表示无选择模式；singleChoice，值为 1，表示最多可以有一项被选中；multipleChoice，值为 2，表示可以有多项被选中
android:divider	setDivider (Drawable divider)	规定 List 项目之间用某个图形或颜色来分隔，也可以用#rgb，#argb，#rrggbb 或者#aarrggbb 的格式来表示某个颜色
android:dividerHeight	setDividerHeight (int height)	分隔符的高度。若没有指明高度，则用此分隔符固有的高度

ListView 的使用需要以下 3 个元素。

（1）ListVeiw：用来展示列表的 View。

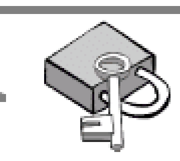
（2）适配器：用来把数据映射到 ListView 上的中介。

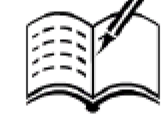
（3）数据：将被映射的字符串、图片或者基本组件。

根据列表的适配器类型，可将列表分为 3 种：ArrayAdapter、SimpleAdapter 和 SimpleCursorAdapter。其中以 ArrayAdapter 最为简单，只能展示一行文字；SimpleAdapter 有最好的扩充性，可以定义出各种各样的布局，可以放上 ImageView（图片），还可以放上 Button（按钮）、CheckBox（复选框）等控件；SimpleCursorAdapter 可以认为是 SimpleAdapter 对数据库的简单结合，可以方便地把数据库的内容以列表的形式展示出来。

6.4.2 列表视图使用实例

在 6.4.1 节中，介绍了 ListView 有 3 种适配器。本节将通过介绍 3 种不同适配器的使用，来介绍列表视图的使用方法。在本节的实例中，在主界面中设计 3 个命令按钮，单击不同的命令按钮，在不同的 Activity 中显示不同的 ListView。在每一个 ListView 中，介绍一种适配器的使用方法。





本实例的开发步骤如下：

（1）创建项目 EX06_4。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8       android:layout_width="fill_parent"
9       android:layout_height="wrap_content"
10      android:text="这是一个列表视图 ListView 的案例"
11      android:textSize="20px"
12      />
13 <Button
14      android:id="@+id/ArrayAdapter"
15      android:layout_width="fill_parent"
16      android:layout_height="wrap_content"
17      android:text="ArrayAdapter"
18      />
19 <Button
20      android:id="@+id/SimpleAdapter"
21      android:layout_width="fill_parent"
22      android:layout_height="wrap_content"
23      android:text="SimpleAdapter"
24      />
25 <Button
26      android:id="@+id/SimpleCursorAdapter"
27      android:layout_width="fill_parent"
28      android:layout_height="wrap_content"
29      android:text="SimpleCursorAdapter"
30      />
31 </LinearLayout>
```

说明：

- 第 2~6 行：定义一个纵向的线性视图，并定义其大小。
- 第 7~12 行：定义一个 TextView 控件，并定义其大小、文本及字体大小。
- 第 13~30 行：定义 3 个 Button 控件，对应 ID 分别为 ArrayAdapter、SimpleAdapter、SimpleCursorAdapter。

（3）为了显示其他 3 个 Activity，依次增加 4 个布局文件：arrayadapter.xml、simpleadapter.xml、simplecursoradapter.xml、list.xml 文件，用于演示 ListView 中 3 种适配器的使用方法，依次编写代码如下：

① arrayadapter.xml 文件

```
1 <?xml version="1.0" encoding="utf-8"?>
```

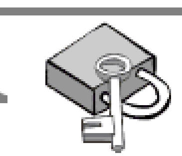
```
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="这是一个 ArrayAdapter 的案例"
11     android:textSize="20px"
12     />
13 <ListView
14     android:id="@+id/arrayList"
15     android:layout_width="fill_parent"
16     android:layout_height="fill_parent"
17     android:divider="#555555"
18     android:dividerHeight="5px"
19     />
20 </LinearLayout>
```

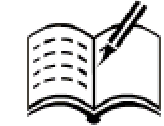
说明：

- 第 2~6 行：定义一个纵向的线性布局，并定义其大小。
- 第 7~12 行：定义一个 TextView 控件，并定义其大小、文本及字体大小。
- 第 13~19 行：定义一个 ListView，并定义其大小，ID 为 arrayList。第 17 行定义 List 项目之间的分隔颜色为#555555，第 18 行定义高度为 5px。

② simpleadapter.xml 文件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
     android:text="这是一个 SimpleAdapter 的案例"
     android:textSize="20px"
     />
<ListView
     android:id="@+id/simpleAdapterList"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:divider="#555555"
     android:dividerHeight="5px"
     />
</LinearLayout>
```

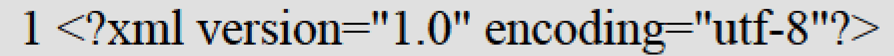




说明：

- ❑ 第 2~6 行：定义一个纵向的线性布局，并定义其大小。
- ❑ 第 7~12 行：定义一个 TextView 控件，并定义其大小、文本及字体大小。
- ❑ 第 13~19 行：定义一个 ListView，并定义其大小，ID 为 arrayList。第 17 行定义 List 项目之间的分隔颜色为#555555，高度为 5px。

③ simplecursoradapter.xml 文件



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="这是一个 SimpleCursorAdapter 的案例"
11     android:textSize="18px"
12     />
13 <ListView
14     android:id="@+id/simpleCursorAdapterList"
15     android:layout_width="fill_parent"
16     android:layout_height="fill_parent"
17     android:divider="#555555"
18     android:dividerHeight="5px"
19     />
20 </LinearLayout>
```

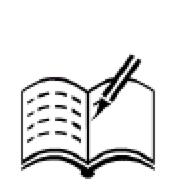
说明：

- ❑ 第 2~6 行：定义一个纵向的线性布局，并定义其大小。
- ❑ 第 7~12 行：定义一个 TextView 控件，并定义其大小、文本及字体大小。
- ❑ 第 13~19 行：定义一个 ListView，并定义其大小，ID 为 arrayList。第 17 行定义 List 项目之间的分隔颜色为#555555，第 18 行定义高度为 5px。

④ list.xml 文件

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:orientation="horizontal"     android:layout_width="fill_parent"
5     android:layout_height="fill_parent">
6     <TextView
7         android:id="@+id/name"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        />
11     <TextView
12         android:id="@+id/phone"
```

```
13        android:layout_width="fill_parent"
14        android:layout_height="wrap_content"
15        android:gravity="right"
16        />
17    </LinearLayout>
```



Note

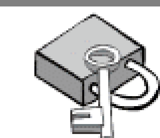
说明：

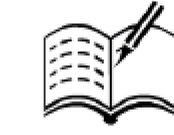
本布局文件主要用于在 simpleadapter.xml、simplecursoradapter.xml 中显示每一个 item 的数据。

- ❑ 第 2~5 行：定义一个横向的线性布局，并定义其大小。
- ❑ 第 6~10 行：定义一个 TextView 控件，并定义其大小，其 ID 为 name。
- ❑ 第 11~16 行：定义一个 TextView 控件，并定义其大小，其 ID 为 phone。

（4）修改主 Activity 的类文件 FirstActivity.java，在本类中，主要通过单击不同的命令按钮，显示不同的 Activity。编写代码如下：

```
package wyq.Ex06_4;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    Button bt_ArrayAdapter;
    Button bt_SimpleAdapter;
    Button bt_SimpleCursorAdapter;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bt_ArrayAdapter=(Button)findViewById(R.id.ArrayAdapter);
        bt_SimpleAdapter=(Button)findViewById(R.id.SimpleAdapter);
        bt_SimpleCursorAdapter=(Button)findViewById(R.id.SimpleCursorAdapter);
        bt_ArrayAdapter.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent();
                intent.setClass(FirstActivity.this, arrayAdapter.class);
                startActivity(intent);
            }
        });
        bt_SimpleAdapter.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent();
                intent.setClass(FirstActivity.this, simpleAdapter.class);
```






```
34              startActivity(intent);
35          }
36      });
37      bt_SimpleCursorAdapter.setOnClickListener(new Button.OnClickListener()
38      {
39          @Override
40          public void onClick(View v) {
41              Intent intent=new Intent();
42              intent.setClass(FirstActivity.this, simpleCursorAdapter.class);
43              startActivity(intent);
44          }
45      });
46  }
47 }
```

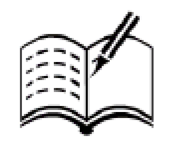
说明：

- 第 9、10 行：声明 3 个 Button 类对象分别为 bt_ArrayAdapter、bt_SimpleAdapter、bt_SimpleCursorAdapter。
- 第 16~18 行：分别获取 ArrayAdapter、SimpleAdapter、SimpleCursorAdapter 控件的引用。
- 第 19~27 行：为 bt_ArrayAdapter 增加单击监听事件，用于显示 arrayAdapter 页面。
- 第 28~36 行：为 bt_SimpleAdapter 增加单击监听事件，用于显示 simpleAdapter 页面。
- 第 37~45 行：为 bt_SimpleCursorAdapter 增加单击监听事件，用于显示 simpleCursorAdapter 页面。

（5）增加 arrayAdapter 类文件 arrayAdapter.java，在这个类中，主要演示 ArrayAdapter 的使用方法。编写代码如下：

```
package wyq.Ex06_4;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
public class arrayAdapter extends Activity {
   /** Called when the activity is first created. */
    ListView listview;
    ArrayAdapter<String> adapter;
    List<String> data = new ArrayList<String>();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.arrayadapter);
        listview=(ListView)findViewById(R.id.arrayList);
        String[] weekList={"星期一","星期二","星期三","星期四","星期五","星期六","星期七"};
```

```
19          adapter=new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,weekList);
20          listview.setAdapter(adapter);
21      }
22 }
```

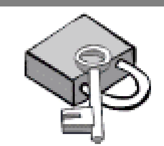

Note

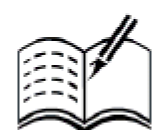
说明：

- ❑ 第 10 行：声明一个 ListView 对象。
- ❑ 第 11 行：声明一个字符串适配器对象。
- ❑ 第 12 行：声明一个字符串列表对象。
- ❑ 第 16 行：设置 Activity 的布局文件为 arrayadapter。
- ❑ 第 17 行：获取 arrayList 控件的引用。
- ❑ 第 18 行：定义 weekList 字符串数组。
- ❑ 第 19 行：创建数组适配器。在创建适配器时，使用的是 Android 系统自带的简单布局 android.R.layout. simple_list_item_1，然后将第 18 行定义的字符串数组传入，作为适配器的数据源。
- ❑ 第 20 行：将 ListView 控件适配器设置为第 19 行所创建的适配器。

（6）增加 simpleAdapter 类文件 simpleAdapter.java 文件。在这个类中，通过将手机的通讯录显示在 ListView 中，来演示 simpleAdapter 的使用方法。为了能够显示程序运行结果，读者需要提前在 Android 虚拟机中增加几条电话号码。编写代码如下：

```
package wyq.Ex06_4;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.provider.ContactsContract.PhoneLookup;
import android.widget.ListView;
import android.widget.SimpleAdapter;
public class simpleAdapter extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.simpleadapter);
    ListView listView=(ListView)findViewById(R.id.simpleAdapterList);
    Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null,
        null, null, null);
    startManagingCursor(cursor);
    List<Map<String, Object>> phoneList = new ArrayList<Map<String, Object>>();
    while (cursor.moveToNext())
```





Note

```
24    {   String PhoneInfo="";
25        Map<String, Object> map = new HashMap<String, Object>();
26        int nameFieldColumnIndex = cursor.getColumnIndex(PhoneLookup.DISPLAY_NAME);
27        String name = cursor.getString(nameFieldColumnIndex);
28         map.put("name", name);
29         String contactId = cursor.getString(cursor.getColumnIndex (ContactsContract.Contacts. _ID));
30        Cursor phone = getContentResolver().query(ContactsContract.CommonDataKinds. Phone.
           CONTENT_URI, null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = "
31                  + contactId, null, null);
32            while (phone.moveToNext())
33            {
34                String strPhoneNumber = phone.getString
              (phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
35                PhoneInfo += strPhoneNumber+"\n";
36            }
37            map.put("phone", PhoneInfo);
38            phone.close();
39            phoneList.add(map);
40        }
41        cursor.close();
42        SimpleAdapter listAdapter = new SimpleAdapter(this,phoneList,R.layout.list,
           new String[]{"name","phone"},new int[]{R.id.name,R.id.phone});
43        listView.setAdapter(listAdapter);
44    }
45 }
```

说明：

- 第 18 行：设置 Activity 的布局文件为 simpleadapter。
- 第 19 行：获取 simpleAdapterList 控件的引用。
- 第 20 行：定义游标，用于获取手机的通讯录。在数据处理中，Android 经常会使用 ContentProvider 的方式。ContentProvider 使用 Uri 实例作为句柄的数据封装，可方便地进行数据的增、删、改、查的操作。Android 并不提供所有应用共享的数据存储，采用 ContentProvider，可以提供简单便捷的接口来保持和获取数据，也可以实现跨应用的数据访问。简单地说，Android 通过 ContentProvider 从数据的封装中获取信息。GetContentResolver()函数则是通过 ContentProvider 提供的 URI 接口来获取里面封装的数据。ContactsContract.Contacts.CONTENT_URI 为联系人数据库提供的 URI。ContentProvider 的具体使用方法将在后面章节进行介绍。
- 第 21 行：打开游标访问联系人数据库，该函数的作用是让 Activity 自身来管理游标。
- 第 22 行：定义一个 Map 类型的列表，用于存放从联系人数据库读取出的联系人信息。
- 第 23~40 行：从联系人数据库读取联系人信息。第 23 行，使用游标进行循环，读取联系人信息；第 25 行定义一个哈希表；第 26、27 行获取联系人姓名；第 28

行将联系人姓名放入哈希表中 name 一列；第 29 行获取联系人的 ID；第 30~36 行获取某联系人的联系电话，因为一个人可能有多个联系电话，所以用一个游标进行循环，遍历该联系人的所有联系电话；第 34 行获取每一个联系电话；第 35 行将该联系人的联系电话连接成一个字符串；第 37 行将联系电话放入哈希表的 phone 一列；第 38 行关闭 phone 游标；第 39 行将哈希表放入 phoneList 列表。

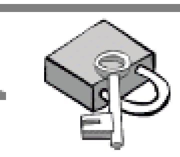
- 第 41 行：关闭外层循环的游标。
- 第 42 行：定义一个 SimpleAdapter 适配器。将上面生成的 phoneList 作为该适配器的数据源，采用 R.layout.list 作为 ListItem 的 XML 实现，String[]{"name","phone"}, new int[]{R.id.name,R.id.phone}定义动态数组与 ListItem 对应的子项。
- 第 43 行：将 ListView 的适配器设置为第 42 行定义的适配器。

（7）增加 simpleCursorAdapter 类文件 simpleCursorAdapter.java。在这个类中，通过将手机的通讯录显示在 ListView 中，来演示 simpleCursorAdapter 的使用方法。编写代码如下：

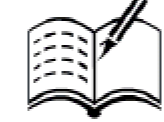
```
1 package wyq.Ex06_4;
2 import android.app.Activity;
3 import android.database.Cursor;
4 import android.os.Bundle;
5 import android.provider.ContactsContract;
6 import android.widget.ListAdapter;
7 import android.widget.ListView;
8 import android.widget.SimpleCursorAdapter;
9 public class simpleCursorAdapter extends Activity {
10      /** Called when the activity is first created. */
11      @Override
12      public void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          setContentView(R.layout.simplecursoradapter);
15          ListView listView=(ListView)findViewById(R.id.simpleCursorAdapterList);
16          Cursor cursor = getContentResolver().query
          (ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
17          startManagingCursor(cursor);
18          ListAdapter listAdapter = new SimpleCursorAdapter
          (this,android.R.layout.simple_expandable_list_item_1,cursor,
          new String[]{ContactsContract.PhoneLookup.DISPLAY_NAME},
          new int[]{android.R.id.text1});
19          listView.setAdapter(listAdapter);
20      }
21 }
```

说明：

- 第 14 行：设置 Activity 的布局文件为 simplecursoradapter。
- 第 15 行：获取 simpleCursorAdapterList 控件的引用。



- ❑ 第 16 行：定义游标，用于获取手机的通讯录。
- ❑ 第 17 行：打开游标访问联系人数据库。
- ❑ 第 18 行：定义一个 SimpleCursorAdapter 适配器。第 2 个参数使用 android 系统提供的布局 android.R.layout.simple_expandable_list_item_1，第 3 个参数用第 16 行定义的游标作为该适配器的数据源，第 4 个参数定义在 ListItem 要显示的内容为联系人的姓名，第 5 个参数定义动态数组与 ListItem 对应的子项。
- ❑ 第 19 行：将 ListView 的适配器设置为第 18 行定义的适配器。



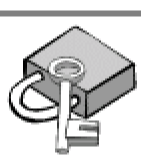
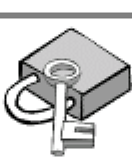
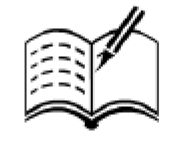
（8）修改 AndroidManifest.xml 文件，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="wyq.Ex06_4"
4       android:versionCode="1"
5       android:versionName="1.0">
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7       <activity android:name=".FirstActivity"
8                 android:label="@string/app_name">
9           <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
11              <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13       </activity>
14       <activity android:name=".arrayAdapter"
15                 android:label="@string/app_name">
16       </activity>
17       <activity android:name=".simpleAdapter"
18                 android:label="@string/app_name">
19       </activity>
20       <activity android:name=".simpleCursorAdapter"
21                 android:label="@string/app_name">
22       </activity>
23     </application>
24     <uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
25     <uses-sdk android:minSdkVersion="5" />
26 </manifest>
```

说明：

- ❑ 第 7~13 行：配置该程序启动的第一个 Activity。
- ❑ 第 14~22 行：配置程序中其他的 Activity。
- ❑ 第 24 行：设置本程序的访问权限。因为本程序要访问手机的电话簿，所以需要相应的权限。

本实例运行结果如图 6-10~图 6-13 所示。

Note



图 6-10　EX06_4 运行结果



图 6-11　ArrayAdapter 适配器

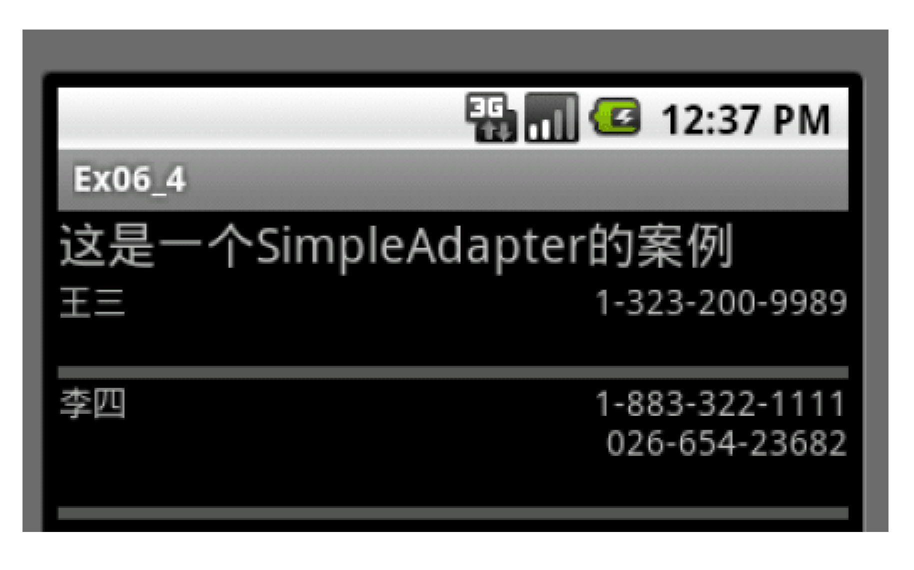


图 6-12　SimpleAdapter 适配器



图 6-13　SimpleCursorAdapter 适配器

6.5　网 格 视 图

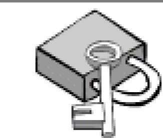
在 6.4 节介绍了列表视图 ListView，本节介绍另外一种视图——网格视图 GridView。

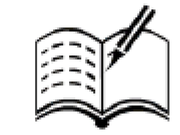
6.5.1　GridView 类简介

GridView 类位于 android.widget 包下。GridView 是一个在平面上可显示多个条目的可滚动的视图组件，该视图可以将其他控件以二维格式显示在表格中。该组件中的条目通过一个 ListAdapter 和该组件进行关联。下面介绍该类一些常用的属性及对应方法，如表 6-4 所示。

表 6-4　GridView 常用属性及对应方法

属 性 名 称	对 应 方 法	说　　明
android:columnWidth	setColumnWidth(int)	设置列的宽度
android:gravity	setGravity (int gravity)	设置此组件中的内容在组件中的位置。可选值有 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、





<table>
<tr><td></td><td></td><td>clip_vertical，可以多选，用“|”分开</td></tr>
</table>

续表

属 性 名 称	对 应 方 法	说　　明
android:horizontalSpacing	setHorizontalSpacing(int)	两列之间的间距
android:numColumns	setNumColumns(int)	列数
android:stretchMode	etStretchMode(int)	缩放模式

6.5.2　网格视图使用实例

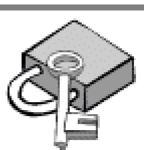
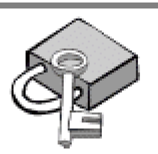
网格视图可以在布局文件中进行配置，也可以通过 Java 代码进行设置。本节的实例将通过在布局文件中进行配置实现。在本实例中，使用 GridView 显示一个丛书列表，并且显示在列表中显示的书目。本实例开发步骤如下：

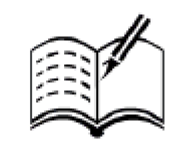
（1）创建项目 EX06_5。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="这是一个 Gridview 网格视图的案例"
     android:textSize="20px"
     />
<TextView
     android:id="@+id/tv"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:textSize="15px"
     android:textColor="#ffffff"
     />
<GridView
     android:id="@+id/gridview"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:numColumns="1"
     android:verticalSpacing="10dp"
     android:horizontalSpacing="10dp"
     android:stretchMode="columnWidth"
/>
</LinearLayout>
```

说明：



- 第 2~6 行：定义一个纵向的线性布局及其大小。
- 第 7~12 行：定义一个 TextView 控件，并定义其大小、文本、字体大小。
- 第 13~19 行：定义一个 TextView 控件，其 ID 为 tv，并定义其大小、字体大小及颜色。
- 第 20~28 行：定义一个 GridView 控件，其 ID 为 gridview，并定义其大小。第 24 行定义 GridView 的列数；第 25 行定义 GridView 的两行之间的间距；第 26 行定义 GridView 的两列之间的间距；第 27 行定义 GridView 的缩放模式。

（3）增加 griditem.xml 文件，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       android:layout_height="wrap_content"
5       android:layout_width="fill_parent"
6       android:orientation="horizontal"
7     >
8     <ImageView
9             android:layout_height="wrap_content"
10            android:id="@+id/ItemImage"
11            android:layout_width="wrap_content"
12            android:layout_centerHorizontal="true"
13      />
14      <TextView
15            android:layout_width="fill_parent"
16            android:layout_height="wrap_content"
17            android:id="@+id/ItemText"
18            android:textSize="15px"
19            android:layout_centerHorizontal="true"
20      />
21 </LinearLayout>
```

说明：

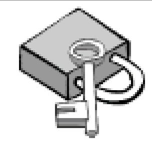
- 第 2~7 行：定义一个水平方向的线性列表，并定义其大小。
- 第 8~13 行：定义一个 ImageView 控件，其 ID 为 ItemImage，对齐方式为水平居中。
- 第 14~20 行：定义一个 TextView 控件，并定义其大小及字体大小，ID 为 ItemText，对齐方式为水平居中。

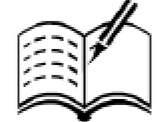
（4）修改主 Acitivity 的类文件 FirstActivity.java，编写代码如下：

```
package wyq.EX06_5;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import android.app.Activity;
```






```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.GridView;
import android.widget.LinearLayout;
import android.widget.SimpleAdapter;
import android.widget.TextView;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private TextView tv;
    private GridView gv;
    private List<Map<String, Object>> bookList ;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        gv=(GridView)findViewById(R.id.gridview);

        int[]picIDs={R.drawable.a,R.drawable.b,R.drawable.c,R.drawable.d,R.drawable.e,R.drawable.f};
        int[]bookIDs={R.string.a,R.string.b,R.string.c,R.string.d,R.string.e,R.string.f};
        int rowCnt=picIDs.length;
        bookList = new ArrayList<Map<String, Object>>();
        for(int i=0;i<rowCnt;i++)
        {
           HashMap<String, Object> map = new HashMap<String, Object>();
           map.put("picCol", picIDs[i]);
           map.put("bookCol", this.getResources().getString(bookIDs[i]));
        bookList.add(map);
        }
        SimpleAdapter ada=new SimpleAdapter(this,bookList,R.layout.griditem,
              new String[]{"picCol","bookCol"},new int[]{R.id.ItemImage,R.id.ItemText});
        gv.setAdapter(ada);
        gv.setOnItemClickListener(new OnItemClickListener()
        {
            @Override
          public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3){
                // TODO Auto-generated method stub
                tv=(TextView)findViewById(R.id.tv);
                LinearLayout l1=(LinearLayout)arg1;
                TextView t1=(TextView)l1.getChildAt(1);
                String str="你选择的书为："+t1.getText().toString();
                tv.setText(str);
            }

        });
    }
```

```
57 }
```

说明：

- 第 20 行：定义 TextView 对象。
- 第 21 行：定义 GridView 对象。
- 第 22 行：定义 HashMap 列表对象。
- 第 28 行：获取 GridView 控件的引用。
- 第 30 行：定义图片 ID 数组。
- 第 31 行：定义书名 ID 列表。
- 第 24~40 行：将图书信息放入 HashMap 列表中。第 38 行 getResources().getString(bookIDs[i])用于获取图书的名字。
- 第 41 行：定义适配器，第二个参数使用上面生成的 List 列表作为适配器的数据源，String[]{"picCol","bookCol"},new int[]{R.id.ItemImage,R.id.ItemText}定义动态数组与 GridItem 对应的子项。
- 第 42 行：设置 GridView 的适配器为第 41 行定义的适配器。
- 第 43~55 行：为 GridView 增加单击监听事件，在 TextView 控件中显示在 GridView 中所选择 Item 的内容。

本实例运行结果如图 6-14 所示，选择 GridView 中一项的结果如图 6-15 所示。



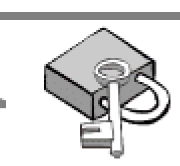
图 6-14　EX06_5 界面

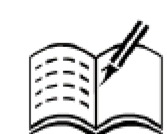


图 6-15　选择 GridView 中的一项

6.6　进度条与滑块

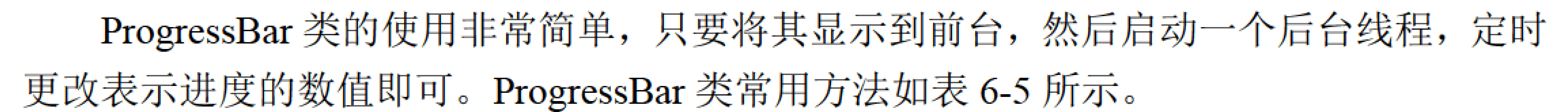
在程序的执行过程中，有些操作可能需要较长的时间，例如某些资源的加载、文件的下载、大量数据的处理等，这时可以使用进度条为用户提供明确的操作结束时间，让用户能够耐心地等待。滑块类似于声音控制条，主要完成与用户的简单交互。本节将介绍进度条 ProgressBar 与滑块 SeekBar 控件的使用。





6.6.1 ProgressBar 类简介

ProgressBar 类位于 android.widget 包下，主要用于显示操作的进度。应用程序可以修改其长度表示当前后台操作的完成情况。因为进度条会移动，所以长时间加载某些资源或者执行某些耗时的操作时，不会使用户界面失去响应。在不确定模式下，可以使进度条循环显示。



ProgressBar 类的使用非常简单，只要将其显示到前台，然后启动一个后台线程，定时更改表示进度的数值即可。ProgressBar 类常用方法如表 6-5 所示。

表 6-5 ProgressBar 类常用方法

方 法	说 明
getMax()	返回进度条范围的上限
getProgress()	返回进度
getSecondaryProgress()	返回次要进度
incrementProgressBy(int diff)	指定增加的进度
isIndeterminate()	指示进度条是否在不确定模式下
setIndeterminate(boolean indeterminate)	设置不确定模式
setVisibility(int v)	设置该进度条是否可视

6.6.2 SeekBar 类简介

SeekBar 类继承自 ProgressBar，是用来接收用户输入的控件，类似于拖拉条，可以直观地显示用户需要的数据。SeekBar 不但可以直观地显示数值的大小，还可以为其设置标度。

6.6.3 进度条与滑块使用实例

本节将通过实例介绍进度条与滑块控件的使用。在本实例中，显示滑块、水平进度条与循环进度条，当单击命令按钮时，使滑块控件与水平进度条控件前进，来演示滑块与水平进度条的使用。本实例开发步骤如下：

（1）创建项目 EX06_6。

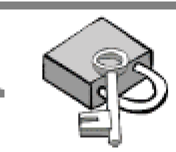
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="这是一个滑块的案例"
```

```
11      />
12 <SeekBar
13      android:layout_width="fill_parent"
14      android:layout_height="wrap_content"
15      android:id="@+id/seekBar"
16      android:max="100"
17      />
18 <TextView
19      android:layout_width="fill_parent"
20      android:layout_height="wrap_content"
21      android:text="这是一个水平进度条的案例"
22      />
23 <ProgressBar
24      android:layout_width="fill_parent"
25      android:layout_height="wrap_content"
26      android:id="@+id/firstBar"
27      android:max="100"
28      style="?android:attr/progressBarStyleHorizontal"
29      />
30 <TextView
31      android:layout_width="fill_parent"
32      android:layout_height="wrap_content"
33      android:text="这是一个循环进度条的案例"
34      />
35 <ProgressBar
36      android:layout_width="wrap_content"
37      android:layout_height="wrap_content"
38      android:id="@+id/secondBar"
39      android:max="100"
40      android:progress="10"
41      style="?android:attr/progressBarStyle"
42      />
43 <Button
44      android:layout_width="100px"
45      android:layout_height="wrap_content"
46      android:id="@+id/bt_Begin"
47      android:text="开始"
48      />
49 </LinearLayout>
```

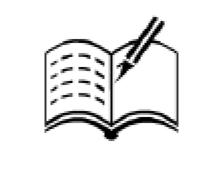
说明：

- 第 2~6 行：定义一个纵向的线性布局，并定义其大小。
- 第 7~11 行：定义一个 TextView 控件，并定义其大小及文本。
- 第 12~17 行：定义一个 SeekBar 滑块控件，并定义其大小，ID 为 seekBar，最大值为 100。
- 第 18~22 行：定义一个 TextView 控件，并定义其大小及文本。
- 第 23~29 行：定义一个 ProgressBar 控件，并定义其大小，ID 为 firstBar，最大值



为 100，其样式为水平进度条。

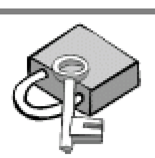
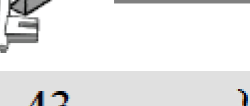
- 第 30~34 行：定义一个 TextView 控件，并定义其大小及文本。
- 第 35~42 行：定义一个 ProgressBar 控件，并定义其大小，ID 为 secondBar，最大值为 100，其样式为循环进度条。



（3）修改主 Acitivity 的类文件 FirstActivity.java，编写代码如下：

```
package wyq.EX06_6;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.SeekBar;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
     private SeekBar seekBar;
     private ProgressBar firstBar;
     private ProgressBar secondBar;
     private Button bt_Begin;
     private int i=0;
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          seekBar=(SeekBar)findViewById(R.id.seekBar);
          firstBar=(ProgressBar)findViewById(R.id.firstBar);
          secondBar=(ProgressBar)findViewById(R.id.secondBar);
          bt_Begin=(Button)findViewById(R.id.bt_Begin);
          bt_Begin.setOnClickListener(new Button.OnClickListener()
          {
               @Override
               public void onClick(View v) {
                    // TODO Auto-generated method stub
                    if(i==0)
                    {
                         firstBar.setVisibility(View.VISIBLE);
                         secondBar.setVisibility(View.VISIBLE);
                    }
                    else if(i<=100)
                    {

                         firstBar.setProgress(i);
                         firstBar.setSecondaryProgress(i+10);
                         secondBar.setProgress(i);
                    }
                    i=i+10;
                    seekBar.setProgress(i);
               }
```


```
43          });
44      }
45 }
```


Note

说明：

- 第 10~13 行：声明 SeekBar、ProgressBar、Button 对象。
- 第 14 行：声明整型变量，用于控制进度条的进度。
- 第 19 行：获取 seekBar 滑块控件的引用。
- 第 20 行：获取 firstBar 进度条控件的引用。
- 第 21 行：获取 secondBar 进度条控件的引用。
- 第 22 行：获取 bt_Begin 按钮控件的引用。
- 第 23~43 行：为 bt_Begin 按钮增加单击监听事件。第 28~32 行，当 i=0 时，设置进度条控件为可视的；第 36 行设置 firstBar 水平进度条的第一进度；第 37 行设置 firstBar 水平进度条的第二进度；第 38 行设置循环进度条的进度；第 41 行设置滑块控件的进度。

本实例运行结果如图 6-16 所示，单击“开始”按钮后结果如图 6-17 所示。



图 6-16　EX06_6 界面



图 6-17　开始后结果

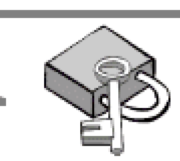
6.7　选　项　卡

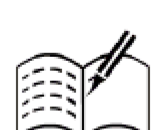
在 Windows 里，用多个选项卡区分不同选项功能的窗口，每个选项卡代表一个活动的区域。在 Android 系统中，也提供了类似的控件 TabHost。本节将介绍 TabHost 控件的使用。

6.7.1　TabHost 类简介

TabHost 类位于 android.widget 包下，用于创建选项卡窗口。TabHost 继承于 FrameLayout，是帧布局的一种，其中可以包含多个布局，然后根据用户的选择显示不同的选项卡窗口。

TabHost 是整个 Tab 的容器，包括两部分：TabWidget 和 FrameLayout。TabWidget 就是每个选项卡的标签，FrameLayout 则是选项卡的内容。





6.7.2　选项卡使用实例

选项卡的实现有以下两种方式。

第一种：继承 TabActivity，从 TabActivity 中用 getTabHost()方法获取 TabHost。各个 Tab 中的内容在布局文件中定义即可。

第二种：不继承 TabActivity，在布局文件中定义 TabHost 即可，但是 TabWidget 的 ID 必须是@android:id/tabs，FrameLayout 的 ID 必须是@android:id/tabcontent，TabHost 的 ID 可以自定义。

本节将通过实例介绍选项卡的两种实现方法。本实例开发步骤如下：

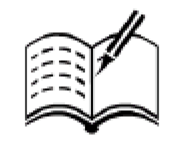
（1）创建项目 EX06_7。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:text="这是一个 TabHost 选项卡控件的案例"
11      />
12 <Button
13     android:layout_width="fill_parent"
14     android:layout_height="wrap_content"
15     android:id="@+id/firstBt"
16     android:text="使用继承 TabActivity 的方式实现 TabHost"
17     android:textSize="15px"
18     />
19 <Button
20     android:layout_width="fill_parent"
21     android:layout_height="wrap_content"
22     android:id="@+id/secondBt"
23     android:text="使用布局文件中定义 TabHost 的方式实现 TabHost"
24     android:textSize="15px"
25     />
26 </LinearLayout>
```

说明：

- ❑ 第 2~6 行：定义一个纵向的线性布局，并定义其大小。
- ❑ 第 7~11 行：定义一个 TextView 控件，并定义其大小及文本。
- ❑ 第 12~18 行：定义一个 Button 控件，并定义其大小、文本及字体大小，ID 为 firstBt。
- ❑ 第 19~25 行：定义一个 Button 控件，并定义其大小、文本及字体大小，ID 为

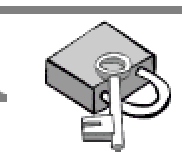

Note

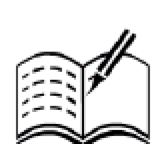
secondBt。

（3）在本实例中，要采用两种方式实现 TabHost，所以需要增加 tabactivity.xml 与 tabxml.xml 两个布局文件。

① 编写 tabactivity.xml，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:id="@+id/firstTab"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center_horizontal"
        android:orientation="vertical">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="这是第一个选项卡"
            android:textSize="20px"/>
    </LinearLayout>
    <LinearLayout
        android:id="@+id/secondTab"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center_horizontal"
        android:orientation="vertical">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="这是第二个选项卡"
            android:textSize="20px"/>
    </LinearLayout>
    <LinearLayout
        android:id="@+id/thirdTab"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center_horizontal"
        android:orientation="vertical">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="这是第三个选项卡"
            android:textSize="20px"/>
    </LinearLayout>
</FrameLayout>
```





说明：

- 第 2~5 行：定义一个帧布局，并定义其大小。
- 第 6~11 行：在帧布局中定义一个纵向的线性布局，并定义其大小，对齐方式为水平居中，ID 为 firstTab。
- 第 12~16 行：在线性布局中定义一个 TextView 控件，并定义其大小、文本及文本字体大小。
- 第 18~23 行：在帧布局中定义一个纵向的线性布局，并定义其大小，对齐方式为水平居中，ID 为 secondTab。
- 第 24~28 行：在线性布局中定义一个 TextView 控件，并定义其大小、文本及文本字体大小。
- 第 30~35 行：在帧布局中定义一个纵向的线性布局，并定义其大小，对齐方式为水平居中，ID 为 thirdTab。
- 第 36~40 行：在线性布局中定义一个 TextView 控件，并定义其大小、文本及文本字体大小。

② 编写 tabxml.xml，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/hometabs"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="使用布局文件中定义 TabHost 的方式实现 TabHost"
        android:textSize="15px"
        />
    <TabHost
        android:id="@+id/tabhost"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">
            <TabWidget android:id="@android:id/tabs"
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content">
            </TabWidget>
            <FrameLayout android:id="@android:id/tabcontent"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">
                  <TextView
```

```
30                              android:id="@+id/view1"
31                              android:layout_width="fill_parent"
32                              android:layout_height="fill_parent"
33                              android:text="这是第一个选项卡"
34                              android:textSize="20px"/>
35                         <TextView
36                              android:id="@+id/view2"
37                              android:layout_width="fill_parent"
38                              android:layout_height="fill_parent"
39                              android:text="这是第二个选项卡"
40                              android:textSize="20px"/>
41                         <TextView
42                              android:id="@+id/view3"
43                              android:layout_width="fill_parent"
44                              android:layout_height="fill_parent"
45                              android:text="这是第三个选项卡"
46                              android:textSize="20px"/>
47                  </FrameLayout>
48              </LinearLayout>
49        </TabHost>
50 </LinearLayout>
```

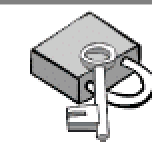

Note

说明：

- 第 2~6 行：定义一个线性纵向布局，并定义其大小，ID 为 hometabs。
- 第 7~12 行：定义一个 TextView 控件，并定义其大小、文本及文本字体大小。
- 第 13~16 行：定义 TabHost 及其大小，ID 为 tabhost。
- 第 17~20 行：定义一个纵向的线性布局，并定义其大小。
- 第 21~25 行：在线性布局中定义水平方向的 TabWidget 控件及其大小，ID 为 tabs。
- 第 26~28 行：定义帧布局及其大小，ID 为 tabcontent。
- 第 29~34 行：定义 TextView 控件及其大小、文本、文本字体大小，ID 为 view1。
- 第 35~40 行：定义 TextView 控件及其大小、文本、文本字体大小，ID 为 view2。
- 第 41~46 行：定义 TextView 控件及其大小、文本、文本字体大小，ID 为 view3。

（4）修改主 Activity 的类文件 FirstActivity.java，在本类中，通过单击不同的命令按钮，显示不同的 Activity。编写代码如下：

```
package wyq.EX06_7;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
     private Button firstBt;
     private Button secondBt;
     @Override
```



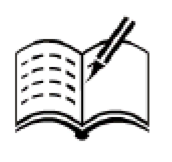

```
12      public void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          setContentView(R.layout.main);
15          firstBt=(Button)findViewById(R.id.firstBt);
16          secondBt=(Button)findViewById(R.id.secondBt);
17          firstBt.setOnClickListener(new Button.OnClickListener()
18          {
19              @Override
20              public void onClick(View v) {
21                  Intent intent=new Intent();
22                  intent.setClass(FirstActivity.this, tabActivity.class);
23                  startActivity(intent);
24                  }
25          });
26          secondBt.setOnClickListener(new Button.OnClickListener()
27          {
28              @Override
29              public void onClick(View v) {
30                  Intent intent=new Intent();
31                  intent.setClass(FirstActivity.this, tabXml.class);
32                  startActivity(intent);
33              }
34          });
35      }
36 }
```

说明：

- ❑ 第 9、10 行：声明 Button 对象。
- ❑ 第 15 行：获取 firstBt 控件的引用。
- ❑ 第 16 行：获取 secondBt 控件的引用。
- ❑ 第 17~25 行：为 firstBt 控件增加单击监听事件，用于显示 tabActivity。
- ❑ 第 26~34 行：为 secondBt 控件增加单击监听事件，用于显示 tabXml。

（5）新建 tabActivity 类文件 tabActivity.java。在这个类中，主要演示使用继承 TabActivity 的方式实现 TabHost 的方法。编写代码如下：

```
package wyq.EX06_7;
import android.app.TabActivity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.TabHost;
public class tabActivity extends TabActivity{
        private TabHost myTabHost;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            myTabHost = this.getTabHost();
            LayoutInflater.from(this).inflate(R.layout.tabactivity, myTabHost.getTabContentView(), true);
            myTabHost.addTab(myTabHost
```



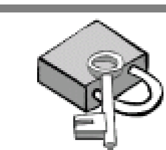

```
14                  .newTabSpec("选项卡 1")
15                  .setIndicator("选项卡 1",getResources().getDrawable(R.drawable.icon))
16                  .setContent(R.id.firstTab));
17              myTabHost.addTab(myTabHost
18                  .newTabSpec("选项卡 2")
19                  .setIndicator("选项卡 2",getResources().getDrawable(R.drawable.icon))
20                  .setContent(R.id.secondTab));
21              myTabHost.addTab(myTabHost
22                  .newTabSpec("选项卡 3")
23                  .setIndicator("选项卡 3",getResources().getDrawable(R.drawable.icon))
24                  .setContent(R.id.thirdTab));
25      }
26 }
```

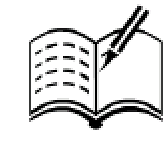
说明：

- 第 7 行：声明 TabHost 对象。
- 第 11 行：获取该 Activity 用于容纳 tab 的 TabHost 对象。
- 第 12 行：LayoutInflater 类用来查找 layout 下 xml 布局文件，并且实例化。将 tabactivity 布局的内容嵌入到 tabhost.getTabContentView()所返回的 FrameLayout 中。
- 第 13~24 行：给 myTabHost 增加 3 个选项卡。newTabSpec("选项卡 1")返回一个 TabHost.TabSpec 对象，用于标识一个选项卡的 Tag；setIndicator("选项卡 1",getResources().getDrawable(R.drawable.icon)) 显示选项卡上的文字；setContent(R.id.firstTab)指定选项卡的内容，参数必须为 ID，比如控件的 ID 或者 layout 的 ID。

（6）增加 tabXml 类文件 tabXml.java。在这个类中，主要演示使用在布局文件中定义 TabHost 的方式实现 TabHost 的方法。编写代码如下：

```
package wyq.EX06_7;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.TabWidget;
public class tabXml extends Activity{
    @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.tabxml);
            TabHost tabHost = (TabHost) findViewById(R.id.tabhost);
            tabHost.setup();
            tabHost.addTab(tabHost
                    .newTabSpec("tab1")
                    .setIndicator("tab1",getResources().getDrawable(R.drawable.icon))
                    .setContent(R.id.view1));
            tabHost.addTab(tabHost
```






```
18                    .newTabSpec("tab2")
19                    .setIndicator("tab2",getResources().getDrawable(R.drawable.icon))
20                    .setContent(R.id.view2));
21            tabHost.addTab(tabHost
22                    .newTabSpec("tab3")
23                    .setIndicator("tab3",getResources().getDrawable(R.drawable.icon))
24                    .setContent(R.id.view3));
25     }
26 }
```

说明：

- 第 11 行：声明一个 TabHost 对象，并获取 tabhost 控件的引用。
- 第 12 行：初始化 TabHost 容器。
- 第 13~24 行：为 tabhost 增加 3 个选项卡。

（7）修改 AndroidManifest.xml 文件，增加 tabActivity 与 tabXml 两个 Activity 的配置。在 AndroidManifest.xml 文件中，增加如下代码：

```
<activity android:name=".tabActivity" android:label="@string/app_name"></activity>
<activity android:name=".tabXml" android:label="@string/app_name"> </activity>
```

本实例运行结果如图 6-18~图 6-20 所示。



图 6-18 EX06_7 界面



图 6-19 TabActivity 实现选项卡



图 6-20 布局文件实现选项卡

6.8 画 廊 控 件

现在手机除了可以进行通信外，还有丰富的娱乐功能，如照相、查看图片等。苹果手机曾经因为其丰富的娱乐功能吸引了不少手机粉丝，例如在查看图片时，点击后一张图片后前一张图片就会往前移动，而点击的图片就会突出显示，也可以触摸拖动图片，任意选择想要的图片突出显示。那么在 Android 上同样可以实现此效果。画廊控件就是一种具有此酷炫效果及使用方法简单的图片浏览控件，是设计相册和图片浏览的首选控件。本节将介绍画廊控件 Gallery 的使用。

6.8.1 Gallery 类简介

Gallery 是一种水平滚动的列表，用来显示图片等资源，可以使图片在屏幕上通过手指的滑动来显示。该类位于 android.widget 包下，其常用的属性如表 6-6 所示。



表 6-6　Gallery 常用属性

属 性 名 称	描　　述	
android:animationDuration	设置布局变化时动画的转换所需的时间（毫秒级）。仅在动画开始时计时。该值必须是整数，如 100	
android:gravity	指定在对象的 X 轴和 Y 轴上如何放置内容。可以指定以下常量中的一个或多个（使用“|”分隔）	
	Constant	Description
	top	紧靠容器顶端，不改变其大小
	bottom	紧靠容器底部，不改变其大小
	left	紧靠容器左侧，不改变其大小
	right	紧靠容器右侧，不改变其大小
	center_vertical	垂直居中，不改变其大小
	fill_vertical	垂直方向上拉伸至充满容器
	center_horizontal	水平居中，不改变其大小
	Fill_horizontal	水平方向上拉伸使其充满容器
	center	居中对齐，不改变其大小
	fill	在水平和垂直方向上拉伸，使其充满容器
	clip_vertical	垂直剪切（当对象边缘超出容器时，将上下边缘超出的部分剪切掉）
	clip_horizontal	水平剪切（当对象边缘超出容器时，将左右边缘超出的部分剪切掉）
android:spacing	设置图片之间的间距	
android:unselectedAlpha	设置未选中的条目的透明度（Alpha）。该值必须是 float 类型，如 1.2	

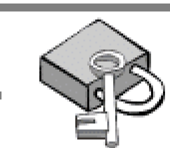
6.8.2　画廊控件使用实例

本节通过一个实例介绍画廊控件的使用方法。在本实例中首先将要显示的控件存放到 BaseAdapter 中，然后将此 BaseAdapter 同 Gallery 控件进行显示。本实例的开发步骤如下：

（1）创建项目 EX06_8。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="这是一个 Gallery 画廊控件的案例"
```





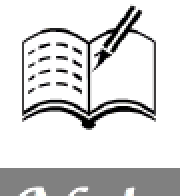

```
11      />
12 <Gallery
13      android:id="@+id/mygallery"
14      android:layout_width="fill_parent"
15      android:layout_height="fill_parent"
16      />
17 </LinearLayout>
```

说明：

- 第 2~6 行：定义一个纵向的线性布局及其大小。
- 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- 第 12~16 行：定义一个 Gallery 控件及其大小，其 ID 为 mygallery。

（3）修改主 Activity 的类文件 FirstActivity.java。在本类中，主要实现 BaseAdapter，使用 Gallery 显示图片。编写代码如下：

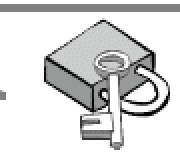
```
package wyq.EX06_8;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private Gallery mGallery;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mGallery = (Gallery)findViewById(R.id.mygallery);
        mGallery.setAdapter(new ImageAdapter(this));
        mGallery.setOnItemClickListener(new OnItemClickListener() {
            @Override
           public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,   long arg3)
            {
                Toast.makeText(FirstActivity.this, "点击了第"+(arg2+1)+"张图片",
                Toast.LENGTH_LONG).show();
            }
        });
    }
    private class ImageAdapter extends BaseAdapter{
        private Context mContext;
        private Integer[] mImage = {
```

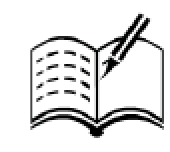

Note

```
33                  R.drawable.simple1,
34                  R.drawable.simple2,
35                  R.drawable.simple3,
36                  R.drawable.simple4,
37                  R.drawable.simple5,
38                  R.drawable.simple6,
39                  R.drawable.simple7,
40                  R.drawable.simple8,
41                  R.drawable.simple9,
42                  R.drawable.simple10
43          };
44          public ImageAdapter(Context c){
45              mContext = c;
46          }
47          @Override
48          public int getCount() {
49              return mImage.length;
50          }
51          @Override
52          public Object getItem(int position) {
53              return position;
54          }
55          @Override
56          public long getItemId(int position) {
57              return position;
58          }
59          @Override
60          public View getView(int position, View convertView, ViewGroup parent) {
61              // TODO Auto-generated method stub
62              ImageView i = new ImageView (mContext);
63              i.setImageResource(mImage[position]);
64              i.setScaleType(ImageView.ScaleType.FIT_XY);
65              i.setLayoutParams(new Gallery.LayoutParams(300, 300));
66              return i;
67          }
68      };
69  }
```

说明：

- 第 15 行：声明一个 Gallery 对象。
- 第 20 行：获取 mygallery 控件的引用。
- 第 21 行：为 mGallery 设置适配器为第 30 行定义的 ImageAdapter 类的对象。
- 第 22~28 行：为 mGallery 控件设置单击监听事件。第 26 行为在屏幕上显示提示内容。
- 第 30 行：定义 ImageAdapter 类，继承于 BaseAdapter 类。
- 第 32~43 行：声明一个整数数组，存放要显示的图片 ID。
- 第 48~50 行：定义 getCount()函数，获取该适配器中图片的数量。





- 第 52~54 行：定义 getItem()函数。
- 第 56~58 行：定义 getItemId()函数。
- 第 60~67 行：定义 getView()函数，用于显示相应位置的图片。
- 第 62 行：声明一个 ImageView 控件。
- 第 63 行：设置 ImageView 的图片资源 ID 为该 ImageView 显示的内容。
- 第 64 行：控制图片适合 ImageView 的大小，拉伸图片（不按比例）以填充 View 的高度和宽度。
- 第 65 行：设置 ImageView 的布局参数。

本实例运行结果如图 6-21 所示，点击所浏览图片结果如图 6-22 所示。



图 6-21　EX06_8 界面



图 6-22　点击浏览图片

6.9　习　题

1．在 Android 应用程序中，使用自动完成文本框实现以下功能：输入一个文字，显示相应的游戏提示，如图 6-23 所示。

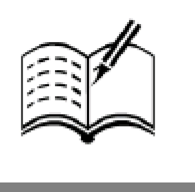
2．设计一个 Android 应用程序，在该程序中使用 Spinner 显示一个下拉列表，并且显示单击的选项，如图 6-24 所示。



图 6-23　游戏列表提示



图 6-24　游戏 Spinner


Note

3．设计一个 Android 应用程序，使用 GridView 显示图书信息。在每条图书信息中显示以下内容，书的图片、名称以及作者，如图 6-25 所示。

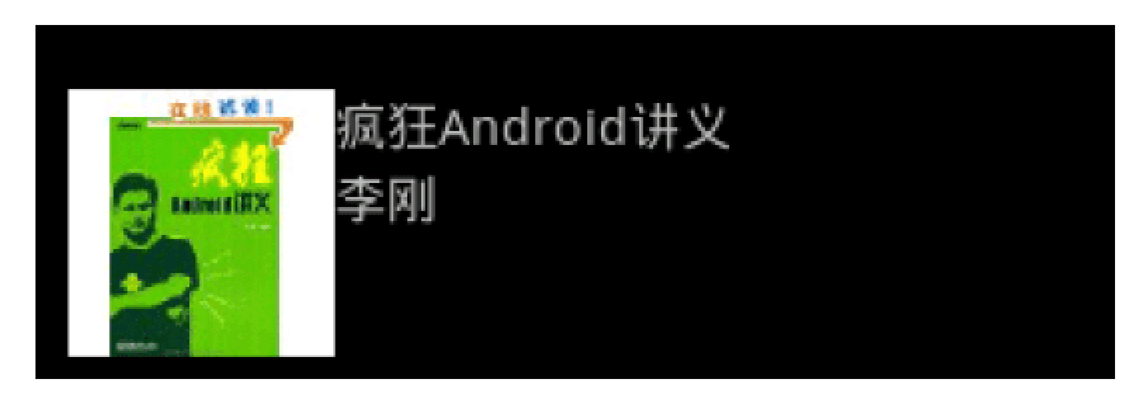


图 6-25　GridView 条目显示方式

4．在 Android 应用程序中，使用 ListView 显示 Android 系统中的文件列表。

5．设计一个 Android 应用程序，模拟后台程序运行进度提示。

6．使用 Gallery 设计一个图片浏览软件，可以浏览手机上的图片文件。

7．在 Android 应用程序中，实现自定义选项卡。

第 7 章 菜单与消息提示

【本章内容】

- ❑ 选项菜单
- ❑ 上下文菜单
- ❑ 对话框
- ❑ 消息提示
- ❑ 状态栏通知

在前面章节中，介绍了开发 Android 应用程序界面经常用到的基本控件与高级控件。但是对于一个软件来说，仅仅有漂亮的控件是不够的，用户体验同样非常重要，方便的操作、有效的互动、及时的提示都可以给软件增色不少。本章将对 Android 应用程序中菜单、对话框、消息提示以及状态栏通知的使用进行介绍。

7.1 选 项 菜 单

对于 Android 应用程序，除了设计人性化的用户界面之外，添加一些菜单可以让应用程序在功能上更加完善。当 Activity 在前台运行时，如果用户按下手机上的 Menu 键，在屏幕底部就会弹出相应的选项菜单，并且 Menu 菜单可以根据用户的需求添加不同的菜单选项。但是这个功能需要开发人员编程实现，如果在开发应用程序时没有实现该功能，则程序运行时按下手机的 Menu 键是不会起作用的。

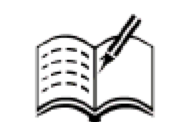
选项菜单可以有文字、图标，也被称做 Icon Menus。默认情况下，每次最多只显示 2 排 3 列 6 个菜单，当菜单选项多于 6 个时，将显示前 5 个选项，而隐藏第 6 项及以后的选项，在第 6 项会出现一个 More 选项，选择该选项才出现第 6 项以及以后的菜单项，这些菜单项也被称做扩展菜单。

7.1.1 选项菜单相关类

开发选项菜单主要用到的类有 Menu、MenuItem 以及 SubMenu。下面对这几个类分别进行简单介绍。

1. Menu 类

一个 Menu 对象代表一个菜单。在 Menu 对象中可以添加菜单项 MenuItem，也可以添


Note

加子菜单 SubMenu。Menu 类常用的方法如表 7-1 所示。

2. MenuItem 类

一个 MenuItem 对象代表一个菜单项，通过 Menu 类的 add()方法，可以将 MenuItem 加入到 Menu 中。MenuItem 类常用的方法如表 7-2 所示。

3. SubMenu 类

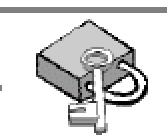
SubMenu 类继承于 Menu 类，一个 SubMenu 对象代表一个子菜单。SubMenu 类中常用的方法如表 7-3 所示。

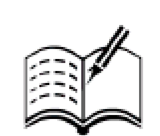
表 7-1 Menu 类的常用方法及说明

方 法	参 数 说 明	说 明
（1）MenuItem add (int groupId, int itemId, int order, CharSequence title) （2）MenuItem add (int groupId, int itemId, int order, int titleRes) （3）MenuItem add (CharSequence title) （4）MenuItem add (int titleRes)	groupId：菜单项所在的组 ID itemId：唯一标识菜单项的 ID order：菜单项的顺序 title：菜单项显示的文本内容 titleRes：String 对象的资源标识符	向 Menu 对象添加一个菜单项，返回 MenuItem 对象
（1）SubMenu addSubMenu (int groupId, int itemId, int order, CharSequence title) （2）SubMenu addSubMenu (int groupId, int itemId, int order, int titleRes) （3）SubMenu addSubMenu (CharSequence title) （4）SubMenu addSubMenu (int titleRes)	groupId：菜单项所在的组 ID itemId：唯一标识菜单项的 ID order：菜单项的顺序 title：菜单项显示的文本内容 titleRes：String 对象的资源标识符	向 Menu 对象添加一个子菜单，返回 SubMenu 对象
MenuItem getItem (int index)		获取菜单中的 MenuItem 对象
MenuItem findItem (int id)		返回指定 ID 的 MenuItem 对象
void removeGroup (int groupId)		如果指定的 ID 组不为空，从菜单中移除该组
void removeItem (int id)		移除指定 ID 的 MenuItem

表 7-2 MenuItem 类的常用方法及说明

方 法	参 数 说 明	说 明
（1）MenuItem setIcon (int iconRes) （2）MenuItem setIcon (Drawable icon)	iconRes：图片资源的标识符 icon：图标 Drawable 对象	设置 MenuItem 的图标
MenuItem setIntent (Intent intent)	intent：与 MenuItem 绑定的 Intent 对象	为 MenuItem 绑定 Intent 对象，当该 MenuItem 被选中时，将会调用 startActivity 方法处理动作相应的 Intent





Note

续表

方　法	参 数 说 明	说　明
MenuItem setOnMenuItemClickListener (MenuItem.OnMenuItemClickListener menuItemClickListener)	menuItemClickListener：监听器	为 MenuItem 设置单击事件监听器
MenuItem setShortcut (char numericChar, char alphaChar)	numericChar：数字快捷键 alphaChar：字母快捷键	为MenuItem 设置数字快捷键和字母快捷键，当按下快捷键或按住 Alt 键的同时按下快捷键将会触发 MenuItem 的单击事件
（1）MenuItem setTitle (int title) （2）MenuItem setTitle (CharSequence title)	title：标题的资源 ID title：标题的名称	为 MenuItem 设置标题
MenuItem setVisible (boolean visible)	visible：true 或者 false	设置 MenuItem 是否显示



表 7-3　SubMenu 类的常用方法及说明

方　法	参 数 说 明	说　明
（1）SubMenu setHeaderIcon (Drawable icon) （2）SubMenu setHeaderIcon (int iconRes)	icon：标题图标 Drawable 对象 iconRes：标题图标资源 ID	设置子菜单的标题图库
（1）SubMenu setHeaderTitle (int titleRes) （2）SubMenu setHeaderTitle (CharSequence title)	titleResult：标题文本的资源 id title：标题文本对象	设置子菜单的标题
（1）SubMenu setIcon (Drawable icon) （2）SubMenu setIcon (int iconRes)	icon：图标 Drawable 对象 iconRes：图标资源 ID	设置子菜单在父菜单中显示的图标

7.1.2　选项菜单和子菜单使用实例

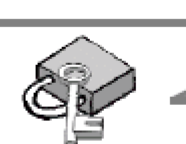
本节将通过实例来说明选项菜单及子菜单的使用方法。在本实例中，首先建立选项菜单和子菜单，当单击某一个菜单选项时，在文本控件中显示该选项的内容。

本实例的开发步骤如下：

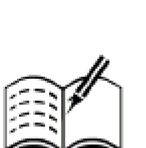
（1）创建项目 EX07_1。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="这是一个选项菜单和子菜单的示例"
    />
<TextView
```



```
13      android:layout_width="fill_parent"
14      android:layout_height="wrap_content"
15      android:id="@+id/tv"
16      android:textSize="20px"
17      />
18 </LinearLayout>
```



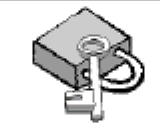
Note

说明：

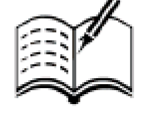
- 第 2~6 行：定义一个纵向的线性布局及其大小。
- 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- 第 12~17 行：定义一个 TextView 控件及其大小、文本字体大小，ID 为 tv，用于显示单击选项菜单的提示信息。

（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

```
package wyq.EX07_1;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
import android.widget.TextView;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private TextView tv;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        SubMenu sub=menu.addSubMenu(Menu.NONE, Menu.FIRST, 0, "发送").
         setIcon(android.R.drawable.ic_menu_send);
        sub.add(Menu.NONE,Menu.FIRST + 6,6,"发送到蓝牙");
        sub.add(Menu.NONE,Menu.FIRST + 7,7,"发送到微博");
        sub.add(Menu.NONE,Menu.FIRST + 8,8,"发送到 E-Mail");
        menu.add(Menu.NONE, Menu.FIRST + 1, 1, "保存")
         .setIcon(android.R.drawable.ic_menu_edit);
        menu.add(Menu.NONE, Menu.FIRST + 2, 2, "帮助")
         .setIcon(android.R.drawable.ic_menu_help);
        menu.add(Menu.NONE, Menu.FIRST + 3, 3, "添加")
         .setIcon(android.R.drawable.ic_menu_add);
        menu.add(Menu.NONE, Menu.FIRST + 4, 4, "详细")
        .setIcon(android.R.drawable.ic_menu_info_details);
        menu.add(Menu.NONE, Menu.FIRST + 5, 5, "退出")
        .setIcon(android.R.drawable.ic_menu_delete);
```



```
28            return true;
29        }
30        @Override
31        public boolean onOptionsItemSelected(MenuItem item)
32        {
33            tv=(TextView)findViewById(R.id.tv);
34            switch (item.getItemId())
35            {
36                case Menu.FIRST :tv.setText("你点击了发送菜单");
37                                break;
38                case Menu.FIRST + 1:tv.setText("你点击了保存菜单");
39                                break;
40                case Menu.FIRST + 2:tv.setText("你点击了帮助菜单");
41                                break;
42                case Menu.FIRST + 3:tv.setText("你点击了添加菜单");
43                                break;
44                case Menu.FIRST + 4:tv.setText("你点击了详细菜单");
45                                break;
46                case Menu.FIRST + 5:tv.setText("你点击了退出菜单");
47                                break;
48                case Menu.FIRST + 6:tv.setText("你点击了发送到蓝牙");
49                                break;
50                case Menu.FIRST + 7:tv.setText("你点击了发送到微博");
51                                break;
52                case Menu.FIRST + 8:tv.setText("你点击了发送到 E-Mail");
53                                break;
54            }
55            return super.onOptionsItemSelected(item);
56        }
57 }
```



Note

说明：

- 第 17~29 行：创建选项菜单。第 19 行定义一个 SubMenu 子菜单对象，并且加入到 menu 中，setIcon 为该菜单选项设置图标，Menu.NONE 表示一个常量 0，用来表示菜单选项的分组；Menu.FIRST 表示常量 1，用来表示菜单选项的 ID。第 20~22 行分别为子菜单对象 sub 增加 3 个菜单选项。第 23~27 行分别为菜单增加 5 个菜单选项，并设置图标。
- 第 31~56 行：重写 onOptionsItemSelected()方法，当 Menu 有命令被选择时，会调用此方法。第 33 行获取 TextView 控件的引用；第 34~54 行根据单击不同的菜单选项，在 TextView 控件中显示不同信息。

本实例运行结果如图 7-1 所示，单击“发送”按钮，结果如图 7-2 所示。

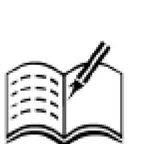

Note



图 7-1　选项菜单



图 7-2　发送子菜单

7.2　上下文菜单

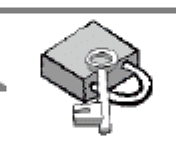
在 7.1 节介绍了选项菜单的使用，本节将介绍上下文菜单（ContextMenu）。上下文菜单继承于 Menu，但是不同于选项菜单，选项菜单服务于某个 Activity，而上下文菜单需要注册到某个 View 对象上。如果在某个 View 对象上注册了上下文菜单，用户可以通过长按该对象大约 2s，打开一个具有相关功能的上下文菜单。

7.2.1　ContextMenu 类简介

上下文菜单不支持快捷键，菜单选项也不能附带图标，但是可以为标题指定图标。ContextMenu 类常用的方法如表 7-4 所示。使用上下文菜单类常用到 Activity 类的成员方法，如表 7-5 所示。

表 7-4　ContextMenu 类常用的方法

方　法	参数说明	说　明
（1）ContextMenu setHeaderIcon (Drawable icon) （2）ContextMenu setHeaderIcon (int iconRes)	iconRes：图片资源的标识符 icon：图标 Drawable 对象	为上下文菜单头设置图标
（1）ContextMenu setHeaderTitle (int titleRes) （2）ContextMenu setHeaderTitle (CharSequence title)	titleRes：标题文本的资源 ID title：标题文本对象	为上下文菜单头的标题栏设置文字
ContextMenu setHeaderView (View view)	view：上下文菜单头要使用的 View	设置 View 到上下文菜单头上。将替代上下文菜单头的图标和标题



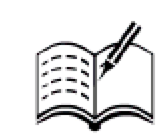


表 7-5 上下文菜单类常用到的 Activity 类的成员方法

方 法 名 称	方 法 说 明
onCreateContextMenu(ContextMenu menu, View v,ContextMenu .ContextMenuInfo menuinfo)	每次为 View 对象呼出上下文菜单
onContextItemSelected(MenuItem item)	当用户选择了上下文菜单选项后调用该方法进行处理
RegisterForContextMenu(View view)	为指定的 View 对象注册一个上下文菜单

7.2.2 上下文菜单使用实例

本节将通过实例介绍上下文菜单 ContextMenu 的使用方法。在本实例中，将在 TextView 和 EditText 控件上绑定上下文菜单，将 TextView 控件中的内容复制到 EditText 控件中，模拟复制/粘贴功能。本实例开发步骤如下：

（1）创建项目 EX07_2。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

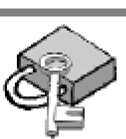
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:id="@+id/tv"
9     android:layout_width="fill_parent"
10    android:layout_height="wrap_content"
11    android:text="这是一个上下文菜单 ContextMenu 的示例"
12    />
13 <EditText
14    android:id="@+id/myEd"
15    android:layout_width="fill_parent"
16    android:layout_height="wrap_content"
17    />
18 </LinearLayout>
```

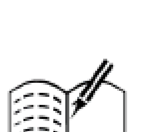
说明：

- ❑ 第 2~6 行：定义一个纵向的线性布局及其大小。
- ❑ 第 7~12 行：定义一个 TextView 控件及其大小、文字。
- ❑ 第 13~17 行：定义一个 EditText 控件及其大小，ID 为 myEd。

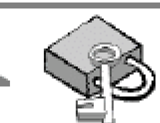
（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

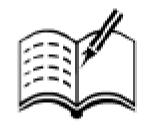
```
package wyq.EX07_2;
import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
```






```
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
public class FirstActivity extends Activity {
        /** Called when the activity is first created. */
    private String tempStr;
    private TextView tv;
    private EditText myEd;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        this.registerForContextMenu(findViewById(R.id.tv));
        this.registerForContextMenu(findViewById(R.id.myEd));
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
    {
        // TODO Auto-generated method stub
        menu.setHeaderIcon(R.drawable.icon);
        if(v==findViewById(R.id.tv))
        {
            menu.add(0,1,0,"复制");
            menu.add(0,2,0,"剪切");
            menu.add(0,3,0,"删除");
        }
        if(v==findViewById(R.id.myEd))
        {
            menu.add(0,4,0,"粘贴");
            menu.add(0,5,0,"删除");
        }
    }
    @Override
    public boolean onContextItemSelected(MenuItem item)
    {
        // TODO Auto-generated method stub
        tv=(TextView)findViewById(R.id.tv);
        myEd=(EditText)findViewById(R.id.myEd);
        switch(item.getItemId())
        {
            case 1:tempStr=tv.getText().toString();
                    break;
            case 2:tempStr=tv.getText().toString();
                    tv.setText("");
                    break;
            case 3:tv.setText("");
                      break;
            case 4:myEd.setText(tempStr);
```






```
55                  break;
56              case 5:myEd.setText("");
57                  break;
58          }
59          return true;
60      }
61 }
```

说明：

- 第 12~14 行：分别定义 String、TextView、EditText 对象。
- 第 19 行：为 TextView 控件绑定上下文菜单。
- 第 20 行：为 EditText 控件绑定上下文菜单。
- 第 23~38 行：重写 onCreateContextMenu()方法，用于创建上下文菜单。
  - 第 26 行：为上下文菜单设置图标。
  - 第 27~32 行：为 TextView 控件的上下文菜单增加菜单项。
  - 第 33~37 行：为 EditText 控件的上下文菜单增加菜单项。
- 第 40~60 行：重写 onContextItemSelected()方法，为每一个菜单项增加方法。当 Menu 有命令被选择时，会调用此方法。

本实例运行结果如图 7-3~图 7-5 所示。



图 7-3　TextView 上下文菜单



图 7-4　EditText 上下文菜单



图 7-5　选择“粘贴”选项

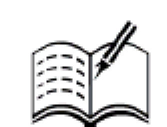
7.3　对　话　框

在用户界面中，除了经常用到菜单之外，对话框也是程序与用户进行交互的主要途径之一。Android 平台下的对话框非常丰富，包括普通对话框、选项对话框、单选/多选对话框、日期和时间对话框等。本节将对 Android 平台下对话框的使用进行介绍。

7.3.1　对话框简介

对话框是 Activity 运行时显示的小窗口。当显示对话框时，当前的 Activity 失去焦点，





Note

对话框获得焦点，与用户进行交流。对话框是 Activity 的一部分，在程序中创建对话框的方法介绍如下。

- onCreateDialog(int)：用于初始化对话框。当使用该回调函数时，Android 系统设置一个 Activity 为对话框的所有者，从而自动管理每个对话框的状态并挂靠到 Activity 上。这样，每个对话框继承这个 Activity 的特定属性。比如，当一个对话框打开时，菜单键显示为这个 Activity 定义的选项菜单，音量键修改 Activity 使用的音频流。
- showDialog(int)：用于显示对话框。当想要显示一个对话框时，调用 showDialog(int id)方法并传递一个唯一标识这个对话框的整数。当对话框第一次被请求时，Android 从 Activity 中调用 onCreateDialog(int id)，这个回调方法被传以与 showDialog(int id)相同的 ID。当创建对话框后，在 Activity 的最后返回这个对象。
- onPrepareDialog(int, Dialog)：在对话框被显示之前，Android 还调用了可选的回调函数 onPrepareDialog(int id, Dialog)。如果想在每一次对话框被打开时改变它的任何属性，可以定义该方法。这个方法在每次打开对话框时被调用，而 onCreateDialog(int) 仅在对话框第一次打开时被调用。如果不定义 onPrepareDialog()，那么这个对话框将保持和上次打开时一样的设置。该方法也被传递对话框的 ID 和在 onCreateDialog()中创建的对话框对象。
- dismissDialog(int)：当准备关闭对话框时，可以通过调用 dismiss()来消除该对话框。如果需要，还可以从这个 Activity 中调用 dismissDialog(int id) 方法，这实际上将为这个对话框调用 dismiss()方法。如果使用 onCreateDialog(int id)方法来管理对话框的状态，每次对话框消除时，这个对话框对象的状态将由该 Activity 保留。如果不再需要这个对象或者要清除该状态，那么应该调用 removeDialog(int id)，这将删除任何内部对象引用，而且如果这个对话框正在显示，它将被消除。

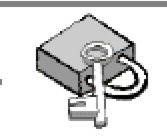
7.3.2 对话框使用实例

在 7.3.1 节中，介绍了对话框的创建方法与过程，本节将通过实例介绍对话框的使用。在本实例中，将通过不同的按钮显示不同的对话框。本实例的开发步骤如下：

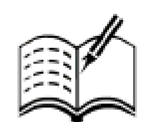
（1）创建项目 EX07_3。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

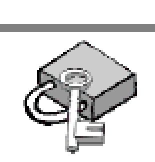
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```



```
    android:text="这是一个对话框的示例"
    />
<Button
    android:id="@+id/bt_showCommonDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示普通对话框"
    />
<Button
    android:id="@+id/bt_showButtonDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示带按钮的对话框"
    />
<Button
    android:id="@+id/bt_showInputDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示输入对话框"
    />
<Button
    android:id="@+id/bt_showListDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示列表对话框"
    />
<Button
    android:id="@+id/bt_showRadioDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示单选按钮对话框"
    />
<Button
    android:id="@+id/bt_showCheckBoxDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示复选框对话框"
    />
<Button
    android:id="@+id/bt_showDatetimePickDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示日期时间对话框"
    />
<Button
    android:id="@+id/bt_showProgressDialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="显示进度条对话框"
```



Note



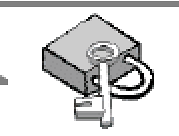

```
59      />
60 <Button
61      android:id="@+id/bt_showMyDialog"
62      android:layout_width="fill_parent"
63      android:layout_height="wrap_content"
64      android:text="显示自定义对话框"
65      />
66 </LinearLayout>
```

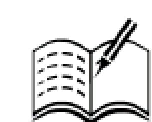
说明：

- 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。
- 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- 第 12~17 行：定义一个 ID 为 bt_showCommonDialog 的 Button 控件及其大小、文本。
- 第 18~23 行：定义一个 ID 为 bt_showButtonDialog 的 Button 控件及其大小、文本。
- 第 24~29 行：定义一个 ID 为 bt_showInputDialog 的 Button 控件及其大小、文本。
- 第 30~35 行：定义一个 ID 为 bt_showListDialog 的 Button 控件及其大小、文本。
- 第 36~41 行：定义一个 ID 为 bt_showRadioDialog 的 Button 控件及其大小、文本。
- 第 42~47 行：定义一个 ID 为 bt_showCheckBoxDialog 的 Button 控件及其大小、文本。
- 第 48~53 行：定义一个 ID 为 bt_showDatetimePickDialog 的 Button 控件及其大小、文本。
- 第 54~59 行：定义一个 ID 为 bt_showProgressDialog 的 Button 控件及其大小、文本。
- 第 60~65 行：定义一个 ID 为 bt_showMyDialog 的 Button 控件及其大小、文本。

（3）新建 login.xml 布局文件，作为自定义对话框的布局，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="用户名：" />
         <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1" />
```





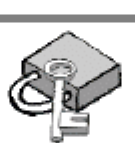
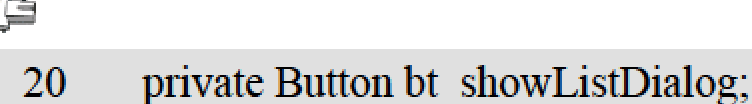
```
20      </LinearLayout>
21      <LinearLayout
22          android:layout_width="fill_parent"
23          android:layout_height="wrap_content"
24          android:gravity="center"
25          android:orientation="horizontal">
26          <TextView
27              android:layout_width="wrap_content"
28              android:layout_height="wrap_content"
29              android:layout_weight="1"
30              android:text="密　　码：" />
31          <EditText
32              android:layout_width="wrap_content"
33              android:layout_height="wrap_content"
34              android:layout_weight="1" />
35      </LinearLayout>
36  </LinearLayout>
```

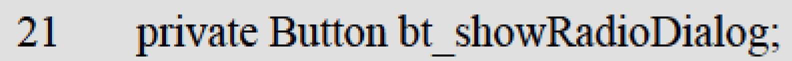
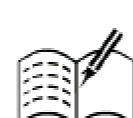
说明：

- 第 2~5 行：定义一个纵向的线性布局，大小为整个屏幕。
- 第 6~10 行：在纵向的线性布局中嵌套一个横向的线性布局，对齐方式为居中。
- 第 11~15 行：定义一个 TextView 控件及其大小、文本。
- 第 16~19 行：定义一个 EditText 控件及其大小。
- 第 21~25 行：在纵向的线性布局中嵌套一个横向的线性布局，对齐方式为居中。
- 第 26~30 行：定义一个 TextView 控件及其大小、文本。
- 第 31~34 行：定义一个 EditText 控件及其大小。

（4）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

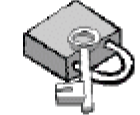
```
package wyq.EX07_3;
import java.util.Calendar;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private Button bt_showCommonDialog;
    private Button bt_showButtonDialog;
    private Button bt_showInputDialog;
```

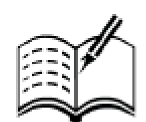


Note

```
private Button bt_showListDialog;
private Button bt_showRadioDialog;
private Button bt_showCheckBoxDialog;
private Button bt_showDatetimePickDialog;
private Button bt_showProgressDialog;
private Button bt_showMyDialog;
final String []arrayHobby={"篮球","足球","羽毛球","乒乓球"};
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
bt_showCommonDialog=(Button)findViewById(R.id.bt_showCommonDialog);
bt_showButtonDialog=(Button)findViewById(R.id.bt_showButtonDialog);
bt_showInputDialog=(Button)findViewById(R.id.bt_showInputDialog);
bt_showListDialog=(Button)findViewById(R.id.bt_showListDialog);
bt_showRadioDialog=(Button)findViewById(R.id.bt_showRadioDialog);
bt_showCheckBoxDialog=(Button)findViewById(R.id.bt_showCheckBoxDialog);
bt_showDatetimePickDialog=(Button)findViewById(R.id.bt_showDatetimePickDialog);
bt_showProgressDialog=(Button)findViewById(R.id.bt_showProgressDialog);
bt_showMyDialog=(Button)findViewById(R.id.bt_showMyDialog);
bt_showCommonDialog.setOnClickListener(new BtClickListener());
bt_showButtonDialog.setOnClickListener(new BtClickListener());
bt_showInputDialog.setOnClickListener(new BtClickListener());
bt_showListDialog.setOnClickListener(new BtClickListener());
bt_showRadioDialog.setOnClickListener(new BtClickListener());
bt_showCheckBoxDialog.setOnClickListener(new BtClickListener());
bt_showDatetimePickDialog.setOnClickListener(new BtClickListener());
bt_showProgressDialog.setOnClickListener(new BtClickListener());
bt_showMyDialog.setOnClickListener(new BtClickListener());
}
class BtClickListener implements OnClickListener
{
    @Override
    public void onClick(View v)
    {
        // TODO Auto-generated method stub
        switch (v.getId())
        {
            case R.id.bt_showCommonDialog: showDialog(1);break;
            case R.id.bt_showButtonDialog: showDialog(2);break;
            case R.id.bt_showInputDialog: showDialog(3);break;
            case R.id.bt_showListDialog: showDialog(4);break;
            case R.id.bt_showRadioDialog: showDialog(5);break;
            case R.id.bt_showCheckBoxDialog: showDialog(6);break;
            case R.id.bt_showDatetimePickDialog: showDialog(7);break;
            case R.id.bt_showProgressDialog: showDialog(8);break;
            case R.id.bt_showMyDialog: showDialog(9);break;
        }
    }
```

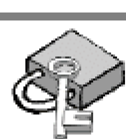


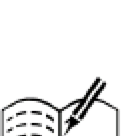


```
}
@Override
protected Dialog onCreateDialog(int id) {
    // TODO Auto-generated method stub
    Dialog alertDialog=null;
    switch(id)
    {
        case 1 :
            alertDialog = new AlertDialog.Builder(this)
                .setTitle("普通对话框")
                .setMessage("这是一个普通对话框")
                .setIcon(R.drawable.icon)
                .create();
                break;
        case 2:
            alertDialog = new AlertDialog.Builder(this)
                .setTitle("确定退出？ ")
                .setMessage("您确定退出程序吗？ ")
                .setIcon(R.drawable.icon)
           .setPositiveButton("确定", new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which) {
                      // TODO Auto-generated method stub
                      finish();
                  }
              })
           .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which) {
                      // TODO Auto-generated method stub
                  }
              })
              .create();
              break;
         case 3:
             alertDialog = new AlertDialog.Builder(this)
                 .setTitle("请输入")
                 .setIcon(R.drawable.icon)
                 .setView(new EditText(this))
                 .setPositiveButton("确定", null)
                 .setNegativeButton("取消", null)
                 .create();
             break;
         case 4:
             alertDialog = new AlertDialog.Builder(this)
                 .setTitle("运动列表")
                 .setIcon(R.drawable.icon)
                 .setItems(arrayHobby, null)
                 .setPositiveButton("确认", null)
```

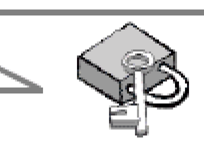


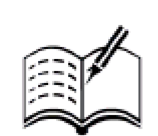
Note






```
                .setNegativeButton("取消", null)
                .create();
            break;
        case 5:
            alertDialog = new AlertDialog.Builder(this)
                .setTitle("你喜欢哪种运动？")
                .setIcon(R.drawable.icon)
                .setSingleChoiceItems(arrayHobby, 0,null)
                .setPositiveButton("确认", null)
                .setNegativeButton("取消", null)
                .create();
            break;
        case 6:
            alertDialog = new AlertDialog.Builder(this)
                .setTitle("你喜欢哪些运动？")
                .setIcon(R.drawable.icon)
                .setMultiChoiceItems(arrayHobby,null,null)
                .setPositiveButton("确认", null)
                .setNegativeButton("取消", null)
                .create();
            break;
        case 7:Calendar c=Calendar.getInstance();
            alertDialog=new DatePickerDialog(this,null,c.get(Calendar.YEAR),
                 c.get(Calendar.MONTH),c.get(Calendar.DAY_OF_MONTH));
            break;
        case 8:
            ProgressDialog pd=new ProgressDialog(this);
                pd.setTitle("下载进度");
                pd.setMax(100);
                pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                pd.setProgress(10);
                pd.setCancelable(true);
            alertDialog=(Dialog)pd;
            break;
        case 9:
            LayoutInflater layoutInflater = LayoutInflater.from(this);
            View loginView = layoutInflater.inflate(R.layout.login, null);
            alertDialog = new AlertDialog.Builder(this)
                .setTitle("用户登录")
                .setIcon(R.drawable.icon)
                .setView(loginView)
                .setPositiveButton("登录", null)
                .setNegativeButton("取消", null)
                .create();
            break;
    }
    return alertDialog;
  }
}
```





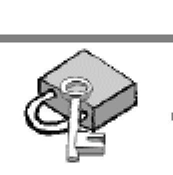
说明：

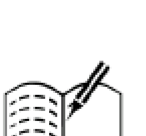
- 第 17~25 行：分别定义 Button 类对象。
- 第 26 行：定义字符数组 arrayHobby。
- 第 31~39 行：获取 Button 控件的引用。
- 第 40~48 行：为 Button 控件增加单击监听事件，setOnClickListener()的参数为继承于 OnClickListener 类的内部类 BtClickListener 对象。
- 第 50~69 行：实现内部类 BtClickListener。在该类中重载了 OnClick()函数，根据单击的 Button 按钮不同，显示不同的对话框。showDialog()函数的参数为对话框的 ID。
- 第 70~163 行：重载 onCreateDialog()函数。根据 ID 的不同，显示不同的对话框。
  - 第 78 行：设置对话框的标题。
  - 第 79 行：设置对话框的消息。
  - 第 80 行：设置对话框的图标。
  - 第 81 行：创建该对话框。
  - 第 88~100 行：为对话框设置按钮，并为该按钮增加单击监听事件。
  - 第 107 行：为对话框设置视图，在该视图中增加一个 EditText 对象。
  - 第 116 行：为对话框设置列表项目。
  - 第 125 行：为对话框设置单选列表项目。
  - 第 134 行：为对话框设置多选列表项目。
  - 第 139 行：声明一个日历对象，并获取当前实例。
  - 第 140 行：定义一个日期对话框，并使用当前年、月、日初始化该日期对话框。
  - 第 143 行：声明一个进度条对话框。
  - 第 144 行：设置进度条对话框的标题。
  - 第 145 行：设置进度条对话框的最大值。
  - 第 146 行：设置进度条对话框的样式。
  - 第 147 行：设置进度条对话框的当前进度。
  - 第 148 行：设置进度条对话框是否可以取消。
  - 第 149 行：将进度条对话框强制转换为 Dialog 对象，并赋值给 alertDialog。
  - 第 152~153 行：获取 login 布局对象。
  - 第 157 行：设置对话框的布局。

本实例运行结果（部分对话框界面）如图 7-6~图 7-8 所示。

7.4　消息提示

在 Android 平台下，除了使用 7.3 节介绍的对话框进行消息提示外，还可以使用 Toast 进行消息提示。本节将介绍 Toast 的使用方法。





Note




图 7-6　主界面



图 7-7　输入对话框



图 7-8　复选按钮对话框

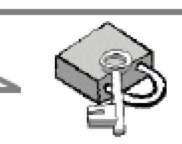
7.4.1　Toast 简介

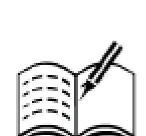
Toast 是一种提供给用户简洁信息的视图，可以创建和显示信息，该视图以浮于应用程序之上的形式呈现给用户。因为它并不获得焦点，即使用户正在输入也不会受到影响。它的目标是尽可能以不显眼的方式，使用户看到提供的信息，例如音量控制提示和设置信息保存成功提示等。

使用该类最简单的方法就是调用一个静态方法 makeText()，来构造需要的一切并返回一个新的 Toast 对象。Toast 类的主要方法如表 7-6 所示。

表 7-6　Toast 类主要方法

方　法	参 数 说 明	说　明
（1）Toast makeText (Context context, int resId, int duration) （2）Toast makeText (Context context, CharSequence text, int duration)	context：使用的上下文。通常是 Activity 对象 resId：要使用的字符串资源 ID duration：该信息的存续期间。值为 LENGTH_SHORT 或 LENGTH_LONG text：Toast 显示的文本	生成一个从资源中取得的包含文本视图的标准 Toast 对象
void setGravity (int gravity, int xOffset, int yOffset)		设置提示信息在屏幕上的显示位置
void setDuration (int duration)	duration：该信息的存续期间。值为 LENGTH_SHORT 或 LENGTH_LONG	设置存续期间
（1）void setText (int resId) （2）void setText (CharSequence s)	resId：Toast 指定的新字符串资源 ID s：Toast 指定的新的文本	之前通过 makeText()方法生成的 Toast 对象的文本内容





Note

7.4.2　Toast 使用实例

本节将通过一个实例来介绍 Toast 的使用方法。在本实例中，单击命令按钮，将会产生一个 Toast 提示。本实例的开发步骤如下：

（1）创建项目 EX07_4。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:text="这是一个 Toast 示例"
11    />
12 <Button
13    android:id="@+id/bt_showToast"
14    android:layout_width="fill_parent"
15    android:layout_height="wrap_content"
16    android:text="显示 Toast"
17    />
18 </LinearLayout>
```

说明：

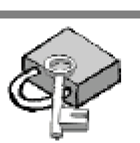
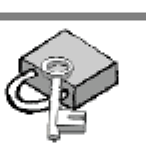
- 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。
- 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- 第 12~17 行：定义一个 ID 为 bt_showToast 的 Button 控件及其大小、文本。

（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

```
package wyq.EX07_4;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Toast;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private Button bt_showToast;
    @Override
```

```
16    public void onCreate(Bundle savedInstanceState) {
17        super.onCreate(savedInstanceState);
18        setContentView(R.layout.main);
19
20        bt_showToast=(Button)findViewById(R.id.bt_showToast);
21        bt_showToast.setOnClickListener(new Button.OnClickListener()
22        {
23            @Override
24            public void onClick(View v) {
25                Toast toast=Toast.makeText(FirstActivity.this,"这是一个带图片的 Toast",
                      Toast.LENGTH_LONG);
26                LinearLayout toastView = (LinearLayout) toast.getView();
27                ImageView imageView = new ImageView(FirstActivity.this);
28                imageView.setImageResource(R.drawable.icon);
29                toastView.addView(imageView);
30                toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
31                toast.show();
32            }
33        });
34    }
35 }
```

说明：

- 第 14 行：声明一个 Button 对象。
- 第 20 行：获取 Button 控件的引用。
- 第 21~33 行：为 Button 控件增加单击监听事件，并重写 onClick(View v)函数。
  - 第 25 行：生成一个 Toast 对象。
  - 第 26 行：获取 Toast 对象的布局。
  - 第 27 行：声明一个 ImageView 对象。
  - 第 28 行：为该 ImageView 对象设置图片资源。
  - 第 29 行：将 ImageView 对象加入到 Toast 视图中。
  - 第 30 行：设置 Toast 对象在屏幕上的显示位置。
  - 第 31 行：显示该 Toast 对象。

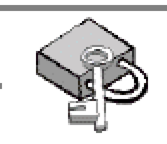
本实例运行结果如图 7-9 和图 7-10 所示。



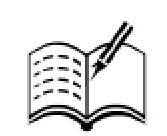
图 7-9　程序主界面



图 7-10　显示 Toast



7.5　状态栏通知



状态栏通知 Notification 是 Android 平台下另外一种消息提示的方式。Notification 位于手机的状态栏（位于屏幕的最上方，通常显示电池电量、信号强度等），用手指按下状态栏并向下拉可以查看状态栏的系统提示消息。

7.5.1　Notification 类简介

Notification 类表示一个持久的通知，可以让应用程序在没有开启的情况下或在后台运行警示用户。它是看不见的程序组件（Broadcast Receiver、Service 和不活跃的 Activity），警示用户有需要注意的事件发生的最好途径。

Notification 类的主要方法如表 7-7 所示。

表 7-7　Notification 类的主要方法

方　法	参 数 说 明	说　明
（1）void cancel (int id) （2）void cancel (String tag，int id)	id：通知的 ID tag：通知的标签	移除一个已经显示的通知，如果该通知是短暂的，会隐藏视图；如果通知是持久的，会从状态栏中移除
void cancelAll ()		移除所有已经显示的通知
（1）void notify (int id，Notification notification) （2）void notify (String tag，int id，Notification notification)	id：应用中通知的唯一标识 notification：一个通知对象，用来描述向用户展示什么信息，不能为空 tag：用来标识通知的字符串，可以为空	提交一个通知，在状态栏中显示。如果拥有相同 ID 的通知已经被提交而且没有被移除，该方法会用新的信息来替换之前的通知
void setLatestEventInfo(Context context，CharSequence contentTitle, CharSequence contentText, PendingIntent contentIntent)	context：上下文环境 contentTitle：状态栏中的大标题 contentText：状态栏中的小标题 contentInten：单击后将发送 PendingIntent 对象	显示在拉伸状态栏中的 Notification 属性，单击后将发送 PendingIntent 对象

创建一个 Notification 的步骤可以简单分为以下 4 步。

（1）通过 getSystemService()方法得到 NotificationManager 对象。

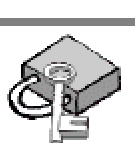
（2）对 Notification 的一些属性进行设置，如内容、图标、标题，对相应 Notification 的动作进行处理等。

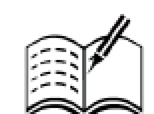
（3）通过 NotificationManager 对象的 notify()方法来执行一个 Notification 的通知。

（4）通过 NotificationManager 对象的 cancel()方法来取消一个 Notificatioin 的通知。

7.5.2　Notification 使用实例

本节将通过一个实例来介绍 Notification 的使用方法。本实例开发步骤如下：





Note

（1）创建项目 EX07_5。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="这是一个 Notification 使用示例"
11     />
12 <Button
13     android:id="@+id/bt_sendNotification"
14     android:layout_width="fill_parent"
15     android:layout_height="wrap_content"
16     android:text="发送 Notification"
17     />
18 </LinearLayout>
```

说明：

- 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。
- 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- 第 12~17 行：定义一个 ID 为 bt_sendNotification 的 Button 控件及其大小、文本。

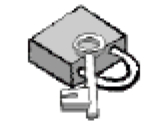
（3）新建 second.xml 布局文件，作为通过 Notification 启动的 Activity 的布局，编写代码如下：

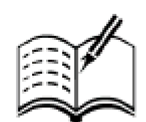
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10     android:text="这是一个通过 Notification 启动的 Activity"
11     />
12 </LinearLayout>
```

说明：

- 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。
- 第 7~11 行：定义一个 TextView 控件及其大小、文本。

（4）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：



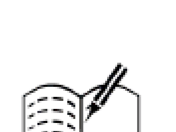



```
1 package wyq.EX07_5;
2
3 import android.app.Activity;
4 import android.app.Notification;
5 import android.app.NotificationManager;
6 import android.app.PendingIntent;
7 import android.content.Intent;
8 import android.os.Bundle;
9 import android.view.View;
10 import android.widget.Button;
11
12 public class FirstActivity extends Activity {
13      /** Called when the activity is first created. */
14      private Button bt_sendNotification = null;
15      private Intent mIntent = null;
16      private PendingIntent mPendingIntent = null;
17      private Notification mNotification = null;
18      private NotificationManager mNotificationManager = null;
19      @Override
20      public void onCreate(Bundle savedInstanceState) {
21           super.onCreate(savedInstanceState);
22           setContentView(R.layout.main);
23           bt_sendNotification=(Button)findViewById(R.id.bt_sendNotification);
24           bt_sendNotification.setOnClickListener(new View.OnClickListener()
25           {
26                @Override
27                public void onClick(View v)
28                {
29                     mNotificationManager = (NotificationManager)getSystemService
                         (NOTIFICATION_SERVICE);
30                     mIntent = new Intent(FirstActivity.this, SecondActivity.class);
31                     mPendingIntent = PendingIntent.getActivity(FirstActivity.this, 0, mIntent, 0);
32                     mNotification = new Notification();
33                     mNotification.icon=R.drawable.icon;
34                     mNotification.tickerText = "实例";
35                     mNotification.defaults = Notification.DEFAULT_ALL;
36                     mNotification.flags = Notification.FLAG_INSISTENT;
37                     mNotification.setLatestEventInfo(FirstActivity.this, "点击查看", "这是一个
                              Notification 示例", mPendingIntent);
38                     mNotificationManager.notify(1, mNotification);
39                }
40           });
41      }
42 }
```

说明：

- ❑ 第 14 行：声明一个 Button 对象。
- ❑ 第 15 行：声明一个 Intent 对象。



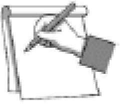


Note

- 第 16 行：声明一个 PendingIntent 对象，PendingIntent 可以理解为延迟执行的 intent，是对 Intent 的一个包装。
- 第 17 行：声明一个 Notification 对象。
- 第 18 行：声明一个 NotificationManager 对象，用来管理 Notification 对象。
- 第 23 行：获取 Button 控件的引用。
- 第 24~40 行：为 Button 控件增加单击监听事件，并重写 onClick(View v)函数。
  - 第 29 行：通过 getSystemService()方法得到 NotificationManager 对象。
  - 第 30 行：定义 Intent 对象，用于启动 SecondActivity 类。
  - 第 31 行：定义 PendingIntent 对象，用于跳转到一个 Activity 组件。
  - 第 32 行：定义 Notification 对象。
  - 第 33 行：设置 Notification 对象的图标。
  - 第 34 行：设置 Notification 对象的提示文字。
  - 第 35 行：设置 Notification 对象的提示方式。常用的常量说明如表 7-8 所示。

表 7-8 提示方式常量及说明

常 量	说 明
DEFAULT_ALL	使用所有默认值，如声音、震动、闪屏等
DEFAULT_LIGHTS	使用默认闪光提示
DEFAULT_SOUND	使用默认提示声音
DEFAULT_VIBRATE	使用默认手机震动



注意

加入手机震动，一定要在 manifest.xml 中加入权限：

<uses-permission android:name="android.permission.VIBRATE" />

以上的效果常量可以叠加，如下所示：

mNotifaction.defaults =DEFAULT_SOUND | DEFAULT_VIBRATE ;

或

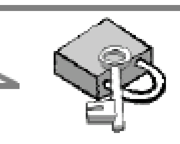
mNotifaction.defaults |=DEFAULT_SOUND

  - 第 36 行：设置 Notification 对象的 Flag 位。常用的常量说明如表 7-9 所示。

表 7-9 Flag 位的常量及说明

常 量	说 明
FLAG_AUTO_CANCEL	该通知能被状态栏的清除按钮清除
FLAG_NO_CLEAR	该通知能被状态栏的清除按钮清除
FLAG_ONGOING_EVENT	将通知放到通知栏的“正在运行”组中
FLAG_INSISTENT	是否一直进行，比如音乐一直播放，直到用户响应

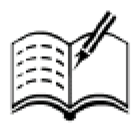
  - 第 37 行：显示在拉伸状态栏中的 Notification 属性，单击后将发送 PendingIntent



对象。

- 第 38 行：提交通知在状态栏中显示。

（5）建立 SecondActivity.java 文件，编写代码如下：



Note

```
package wyq.EX07_5;

import android.app.Activity;
import android.os.Bundle;

public class SecondActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second);
    }
}
```

（6）开发一个新的 Activity 对象 SecondActivity，需要在 AndroidManifest.xml 进行声明，否则系统将无法得知该 Activity 的存在，并进行权限设置。打开 AndroidManifest.xml 文件，在<application>与</application>标记之间加入如下代码：

```
<activity android:name=".SecondActivity"
    android:label="@string/app_name"> </activity>
<uses-permission android:name="android.permission.VIBRATE" />
```

本实例运行结果如图 7-11~图 7-13 所示。

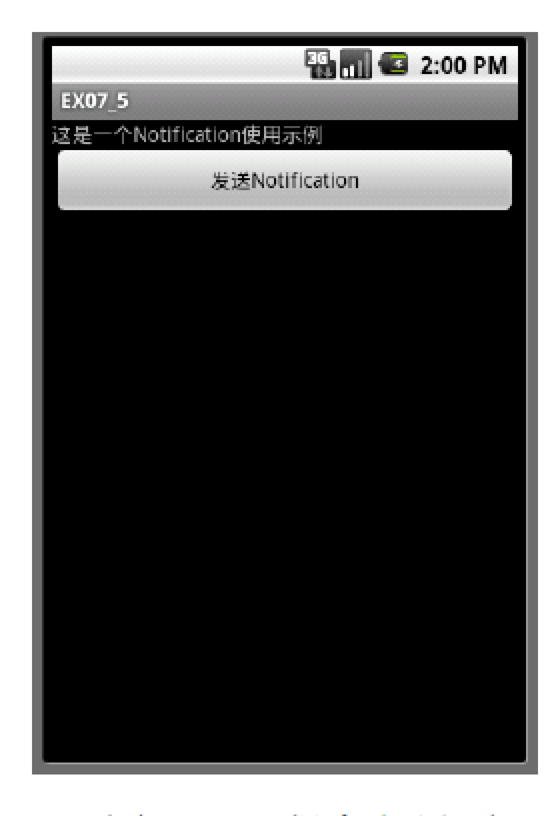


图 7-11　程序主界面



图 7-12　显示 Notification



图 7-13　打开第二个界面

7.6　习　　题

1．在 Android 程序中，实现如图 7-14 所示选项菜单。单击“更多”选项后，显示如图 7-15 所示菜单。

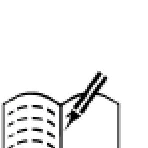

Note



图 7-14　选项菜单



图 7-15　更多选项菜单

2．在 Android 程序中，使用 Alert 对话框，模拟 QQ 的登录界面。

3．设计一个 Android 程序，实现以下功能：

（1）使用 ListView 显示手机中联系人的姓名。

（2）在 ListView 中注册上下文菜单，通过上下文菜单的命令，查看该联系人的详细信息。

（3）通过 ListView 的上下文菜单，对联系人信息进行删除。删除后，使用 Toast 显示提示信息。

4．设计一个 Android 程序，按手机的返回键时，程序在后台运行，程序的图标使用 Notification 在状态栏显示。在状态中单击后，显示该程序的界面。

第 8 章 Android 程序调试

【本章内容】

- DDMS 介绍
- 启动 DDMS
- 使用 DDMS 进行进程管理
- 使用 DDMS 进行文件操作
- 使用模拟器进行控制
- 使用程序日志 LogCat
- 在模拟器或者目标设备上截屏
- 使用手机调试 Android 程序

前面介绍了 Android 应用程序开发的基本组件、控件及消息提示。通过前面几章的介绍，读者可以开发设计一些简单的 Android 应用程序，但是在开发程序的过程中，不可避免地会遇到各种各样的错误。当遇到错误时，开发人员除了要凭借错误信息提示以及经验之外，还可以借助于编译器自身的工具调试程序、排查错误，从而解决问题。在 Android 平台下，开发人员可以借助 DDMS 工具进行程序的调试工作。除此之外，还可以通过手机进行 Android 程序的调试。

8.1 DDMS 介绍

DDMS（Dalvik Debug Monitor Service）是指 Android 开发环境中的 Dalvik 虚拟机调试监控服务，它主要是对系统运行后台日志、系统线程、模拟器状态进行监控，还可以提供以下功能：为测试设备截屏、针对特定的进程查看正在运行的线程以及堆信息、LOGCAT、广播状态信息、模拟电话呼叫、模拟收发短信、发送虚拟地理坐标等。

如果开发人员使用的是安装了 Android 开发工具插件（Android Development Tools Plug-In）的 Eclipse 集成开发环境（Integrated Development Environment，IDE），那么 DDMS 工具已经紧密地融合到了开发环境中。通过 DDMS 视图，可以浏览任何一个在开发机上运行的模拟器实例，并且能够查看通过 USB 连接的 Android 设备。如果没有使用 Eclipse，那么 DDMS 也可以在单独的进程中运行，它位于 Tools 目录下。在这种情况下，DDMS 将运行在自己的进程中。

DDMS 的工作原理为：DDMS 搭建起 IDE 与测试终端（Emulator 或 Connected Device）的连接，它们应用各自独立的端口监听调试信息，DDMS 可以实时监测到测试终端的连接情况。当有新的测试终端连接后，DDMS 将捕捉到终端的 ID，并通过 adb 建立调试器，从而实现发送指令到测试终端的目的。



8.2　启动 DDMS

在 Eclipse 界面的右上角单击添加工具图标，选中 DDMS 后单击 OK 按钮，如图 8-1 所示，即可添加 Eclipse。此时，Eclipse 右上角就会出现 DDMS 图标，单击该图标可开启 DDMS，DDMS 界面如图 8-2 所示。

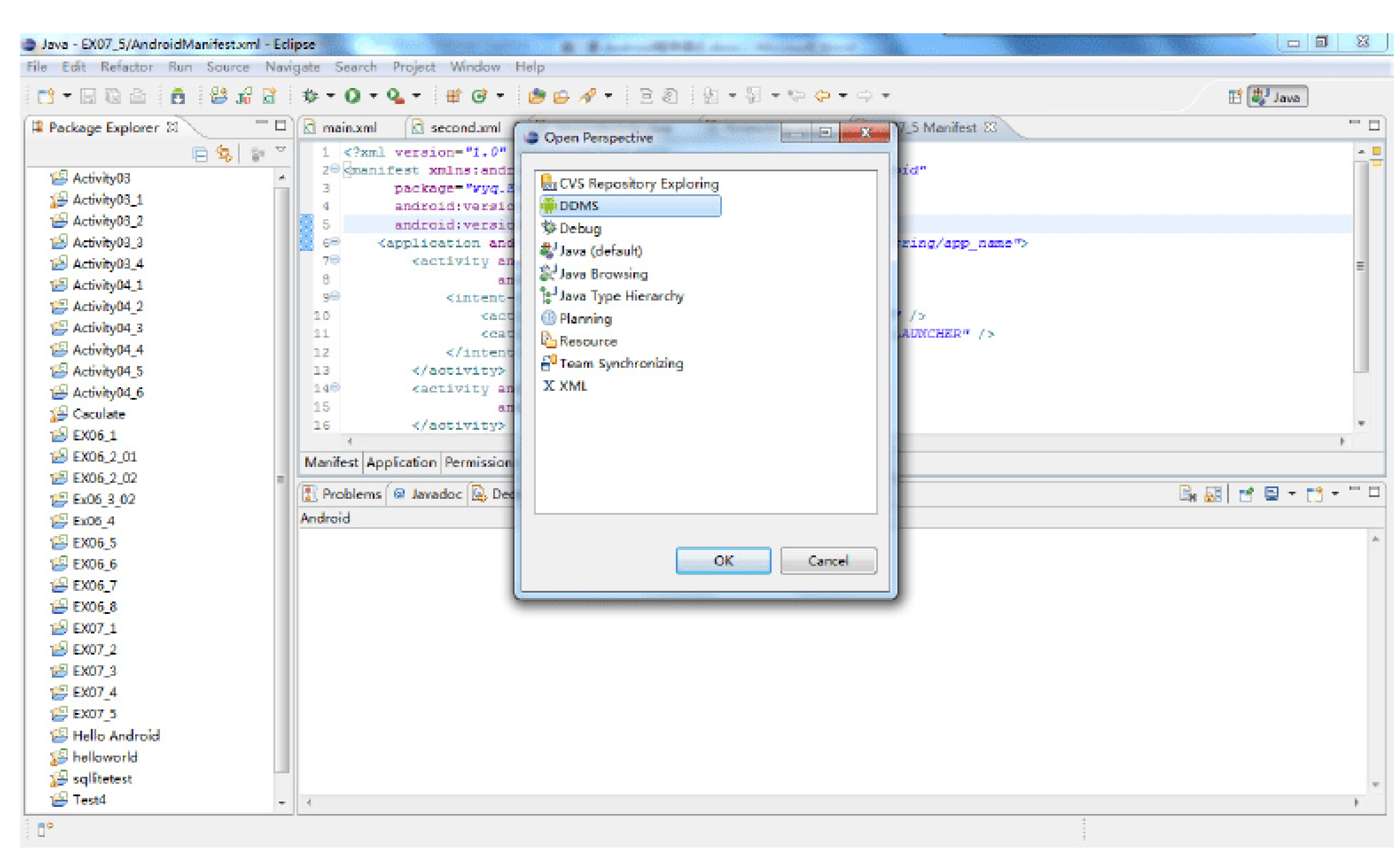


图 8-1　添加 DDMS

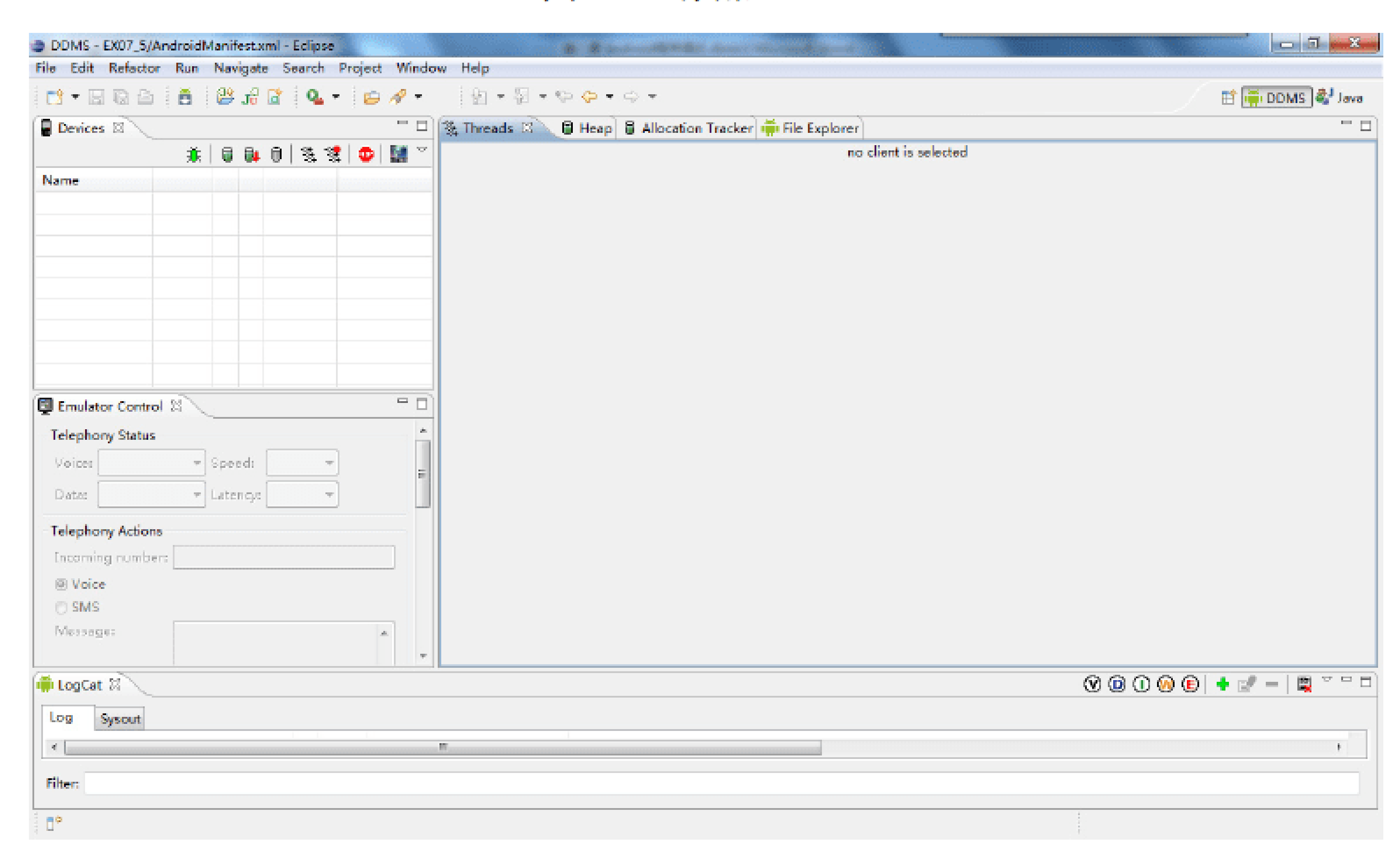
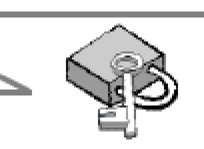
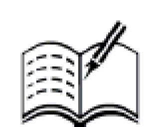


图 8-2　DDMS 界面





DDMS 界面各部分组成的功能简介如下。

（1）Devices：可以查看到所有与 DDMS 连接的模拟器详细信息，以及每个模拟器正在运行的 APP 进程，每个进程最右边相对应的是与调试器连接的端口。

（2）Emulator Control：可以实现对模拟器的控制，如接听电话、根据选项模拟各种不同网络情况、模拟接收 SMS 消息和发送虚拟地址坐标用于测试 GPS 功能等。

① Telephony Status：通过选项模拟语音质量以及信号连接模式。

② Telephony Actions：模拟电话接听和发送 SMS 到测试终端。

③ Location Control：模拟地理坐标或动态的路线坐标变化并显示预设的地理标识，可以通过以下 3 种方式。

- Manual：手动为终端发送二维经纬坐标。
- GPX：通过 GPX 文件导入序列动态变化地理坐标，从而模拟行进中 GPS 变化的数值。
- KML：通过 KML 文件导入独特的地理标识，并以动态形式根据变化的地理坐标显示在测试终端。

（3）LogCat：主要显示日志信息，日志包括 ERROR、WARN、INFO、DEBUG、VERBOSE 共 5 种类型，在 LogCat 中使用其大写首字母来代替，即 V 为所有的信息，D 为 DEBUG 信息，I 为 INFO 信息，W 为 WARN 信息，E 为 ERROR 信息。

通常在代码中使用如下方法来记录日志：Log.v()、Log.d()、Log.i()、Log.w()、Log.e()，具体的参数可以参考 API。在运行项目时，可以通过这里监控到很多系统日志，了解系统的运行状况。

（4）Threads：可以查看某个进程里的所有线程的活动。

（5）Heap：可以监测应用进程使用内存情况。

（6）Allocation Tracker：可以跟踪每个选中的虚拟机的内存分配情况。

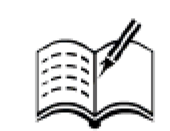
（7）File Explorer：最常用的就是 File Explorer 文件浏览器，通过 File Explorer 可以查看 Android 模拟器中的文件，还可以把文件上传到 Android 手机，或者从手机下载下来，也可以进行删除操作。

8.3　使用 DDMS 进行进程管理

DDMS 非常有用的一个特性是可以同进程打交道。每一个 Android 应用程序都是用其自己的用户 ID 运行在操作系统的单独的 VM（虚拟机）中。

通过 DDMS 左侧的面板，可以查看在设备上运行的 VM 实例，每一个实例均以其包名称作为标识。在 DDMS 的进程管理中，可以进行以下操作。

- 在 Eclipse 中关联（attach）并调试应用程序。
- 监视线程。
- 监视堆。
- 终止进程。


Note

- 强制进行垃圾回收（Garbage Collection，GC）。

1. 向 Android 应用程序关联调试器

虽然大多数情况下会使用 Eclipse 调试参数来运行并调试应用程序，但也可以使用 DDMS 来选择任何需要调试的应用程序，并直接关联和调试。

要为一个进程关联调试器，需要在 Eclipse 工作区中打开对应包的源代码，然后执行以下步骤进行调试。

（1）在模拟器或设备上，确认想要调试的应用程序处于运行状态。

（2）在 DDMS 中，找到应用程序的包，并且单击使其高亮显示。

（3）单击绿色的小虫图标开始调试。

（4）在必要时切换到 Eclipse 的调试视图进行调试。

2. 监视 Android 应用程序的线程活动

可以使用 DDMS 来监视每一个 Android 应用程序的线程活动，步骤如下。

（1）在模拟器或设备上，确认想要监视的应用程序处于运行状态。

（2）在 DDMS 中，找到应用程序的包，并且单击使其高亮显示。

（3）单击带有 3 个箭头的小图标，以显示应用程序的线程。它们将出现在 Threads 选项卡的右侧。默认情况下，这里显示的数据每 4s 进行一次更新。

（4）在 Threads 选项卡中，可以选择某个特定的线程并且单击 Refresh 按钮来深入查看该线程。其中包含的类将会显示在下方区域，结果如图 8-3 所示。

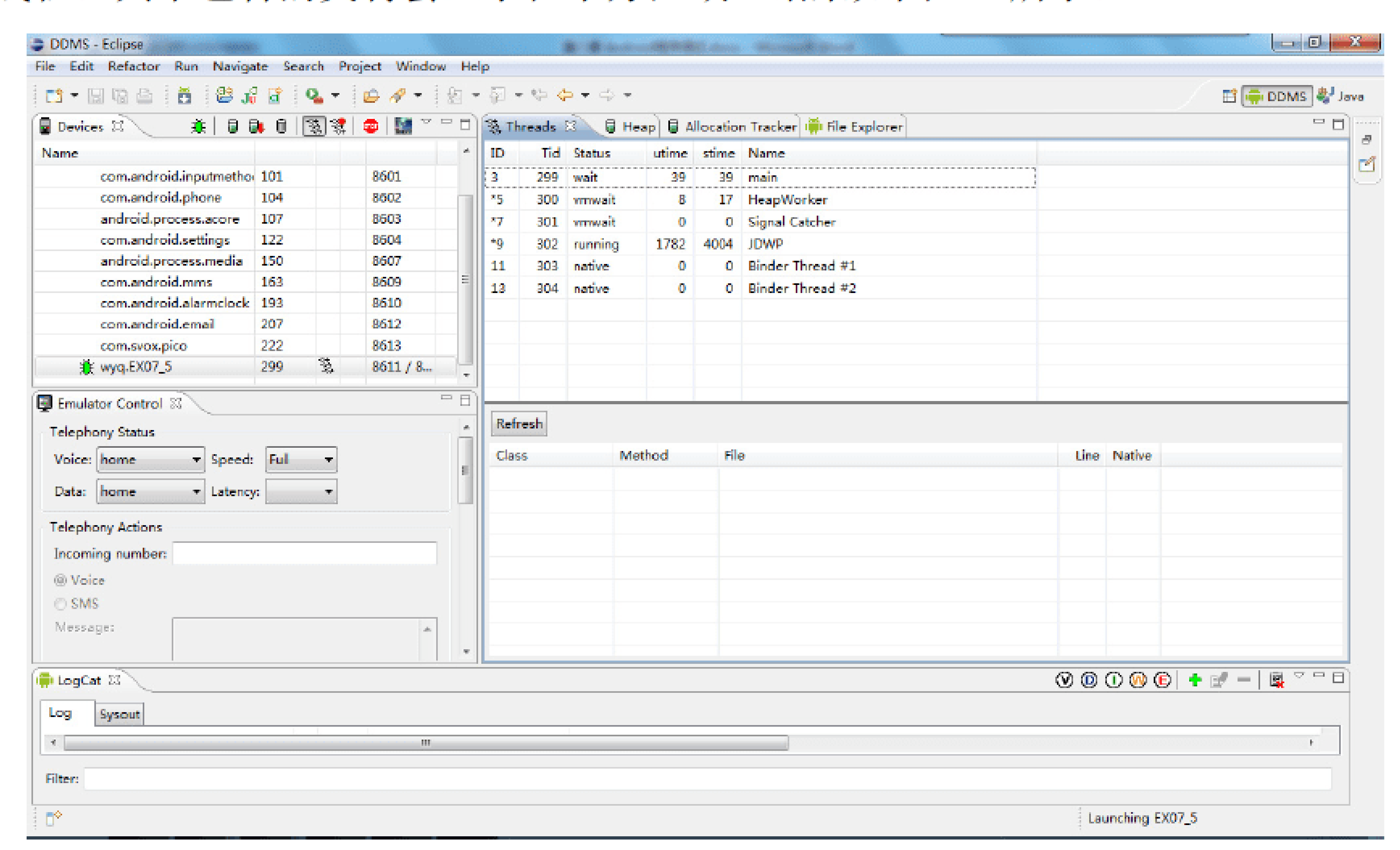
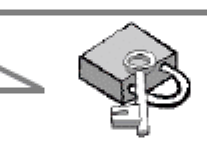


图 8-3 监视 Android 应用程序的线程活动

3. 在 Android 应用程序中触发垃圾回收（GC）

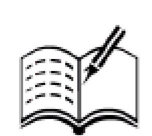
可以使用 DDMS 来强制进行垃圾回收，步骤如下。

（1）在模拟器或设备上，确认想要进行 GC 的应用程序处于运行状态。



（2）在 DDMS 中，找到这个应用程序的包，并且单击使其高亮显示。

（3）展开下拉菜单（单击▽按钮）并且选择 Cause GC 选项。也可以在 Heap 选项卡中执行这一操作。



4. 监视 Android 应用程序的堆活动

可以使用 DDMS 来监视每一个 Android 应用程序的堆统计数据。在每次 GC 后堆的统计数据将进行更新，步骤如下。

（1）在模拟器或设备上，确认想要监视的应用程序处于运行状态。

（2）在 DDMS 中，找到这个应用程序的包，并且单击使其高亮显示。

（3）单击绿色的圆筒图标，以显示该应用程序的堆信息。统计数据将出现在 Heap 选项卡的右侧。这一数据将在每次 GC 后予以更新。也可以通过单击 Heap 选项卡中的 Cause GC 按钮来触发一个 GC 操作。

（4）在 Heap 选项卡中，可以选择特定类型的对象，其使用情况图表将显示在 Heap 选项卡的底部，结果如图 8-4 所示。

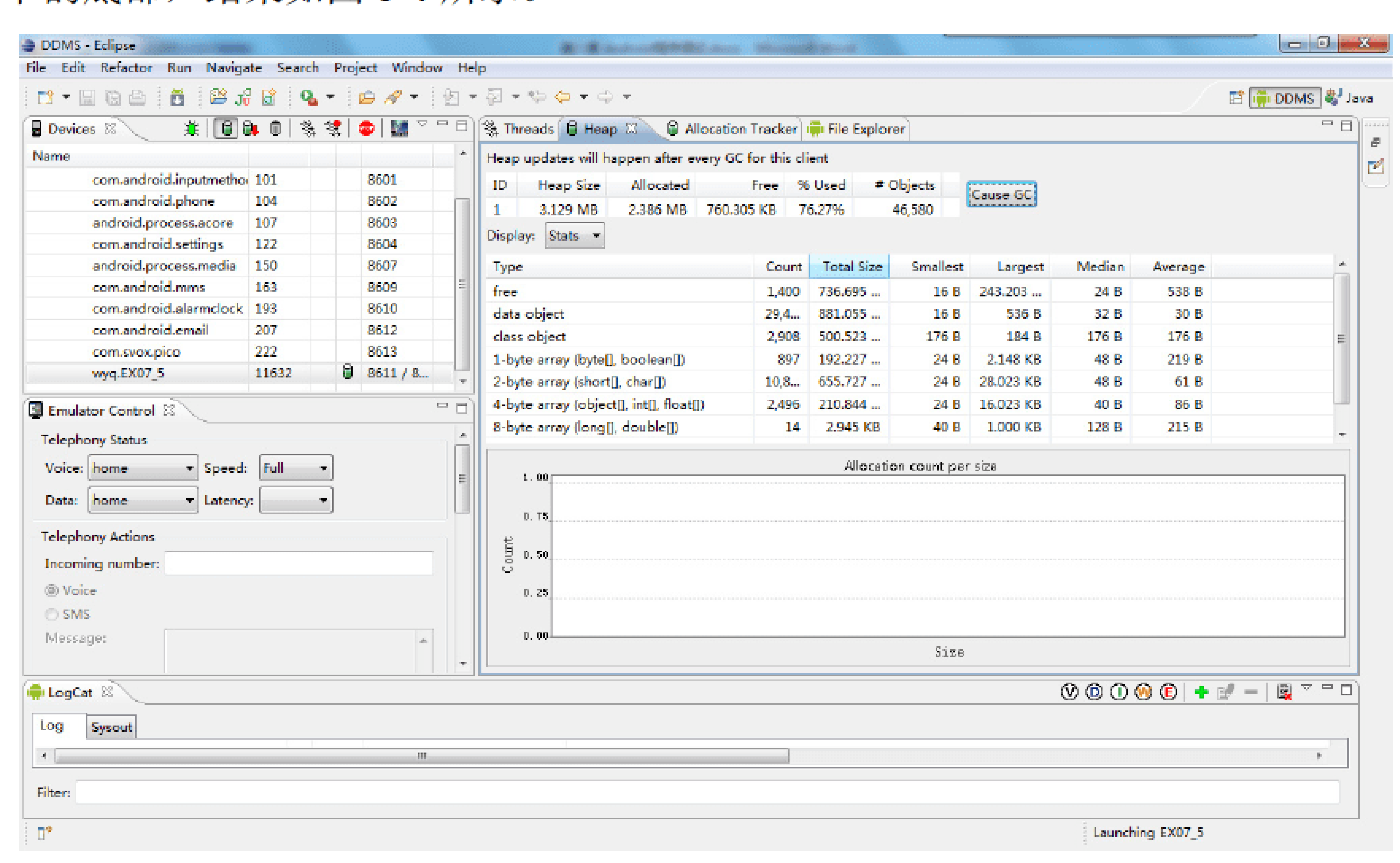


图 8-4　监视 Android 应用程序的堆活动

5. 终止 Android 进程

可以使用 DDMS 来终止一个 Android 应用程序，步骤如下。

（1）在模拟器或者设备上，确认想要终止的应用程序处于运行状态。

（2）在 DDMS 中，找到这个应用程序包，并且单击使其高亮显示。

（3）单击带有红色停止符号的图标，终止该进程。

8.4　使用 DDMS 进行文件操作

开发人员可以使用 DDMS 来查看并操作模拟器或设备上的 Android 文件系统。Android 文件系统中的一些重要区域如表 8-1 所示。

表 8-1　Android 文件系统中的一些重要区域

目　　录	说　　明
\data\data\<packagename>\	应用程序顶层目录，如\data\data\com.androidbook.pettracker
\data\data\<packagename>\shared_prefs\	应用程序共享首选项目录，命名的首选项以 XML 文件的方式进行存储
\data\data\<packagename>\files\	应用程序文件目录
\data\data\<packagename>\cache\	应用程序缓存目录
\data\data\<packagename>\databases\	应用程序数据库目录，如\data\data\com.androidbook.pettracker\databases\test.db
\sdcard\download\	用于存储模拟器上浏览器下载的图像
\data\app\	用于存储第三方 Android 应用程序的 APK 文件

通过 DDMS，可以进行以下操作：

- ❑ 浏览 Android 文件系统。
- ❑ 从模拟器或设备上复制文件。
- ❑ 向模拟器或设备复制文件。
- ❑ 删除模拟器或设备上的文件夹。

1. 浏览 Android 文件系统

要浏览 Android 文件系统，步骤如下。

（1）在 DDMS 中，选择想要浏览的模拟器或设备。

（2）切换到 File Explorer 选项卡，将看到底层显示的目录。

（3）浏览某个文件夹或文件。

2. 从模拟器或设备上复制文件

可以使用文件夹浏览器将模拟器或设备上的文件或文件夹复制到计算机上，步骤如下：

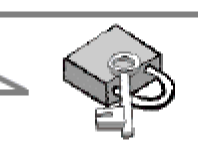
（1）使用文件夹浏览器导航至需要复制的文件或文件夹，单击使其高亮显示。

（2）在文件浏览器的右上角单击 Disk 图标，提取设备中的文件。另外，可以展开图标旁边的下拉菜单，单击按钮，并从中选择 Pull File 来执行这一操作。

（3）输入计算机上用于存放这一文件或文件夹的路径，然后单击 Save 按钮。

3. 向模拟器或设备复制文件

可以使用文件夹浏览器将计算机上的文件复制到模拟器或设备的文件系统中，步骤如下。

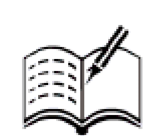


（1）使用文件夹浏览器导航至需要复制文件的文件夹，单击使其高亮显示。

（2）在文件夹浏览器的右上角单击 Phone 图标，向设备中添加文件。另外，可以展开图标旁边的下拉菜单，单击，并从中选择 Push File 来执行这一操作。

（3）选择计算机上待复制的文件，然后单击 Open 按钮。

文件浏览器还支持鼠标拖拽，这也是唯一可以向 Andriod 文件系统中复制文件夹的操作。不过，并不推荐向 Android 文件系统中复制文件夹，因为并没有用于删除它们的选项。但如果拥有许可权限，则可以使用程序来删除这些文件夹。总之，可以从计算机上将一个文件或文件夹拖到文件浏览器中，并在适当的位置释放它。



Note

4. 删除模拟器或设备上的文件夹

可以使用文件浏览器来删除模拟器或设备上的文件（但不能删除文件夹），步骤如下。

（1）使用文件浏览器导航至需要删除的文件，单击该文件使其高亮显示。

（2）在文件浏览器的右上角单击红色的减号图标删除文件。

执行这一操作时需要特别小心，因为没有任何确认提示，文件将被立即删除并且没有办法恢复。

8.5 使用模拟器控制

可以通过 DDMS 的 Emulator Control（模拟器控制）选项卡来操作模拟器实例，在此之前必须选中需要操作的模拟器。可以针对下面的目的使用模拟器控制选项卡。

- 模拟语音来电。
- 模拟 SMS 接收。
- 发送位置坐标。

1. 模拟语音来电

要使用模拟器控制选项卡来模拟语音来电，需执行以下步骤。

（1）在 DDMS 中，选择想要拨打的模拟器。

（2）切换到模拟器控制选项卡。

（3）输入模拟呼入的电话号码，可以包括任意数字、+和#。

（4）选中 Voice 单选按钮。

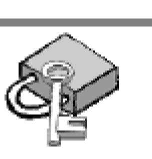
（5）单击 Call 按钮。

（6）模拟器将会接收到呼入并响铃，接听电话即可。

（7）模拟器可以像正常情况一样挂断电话，也可以单击 DDMS 中的 Hang Up 按钮终止通话。过程如图 8-5 和图 8-6 所示。

2. 模拟 SMS 接收

DDMS 提供了最稳定的向模拟器发送 SMS 的方法，其过程同模拟语音来电类似。要使用模拟器控制选项卡模拟发送 SMS，步骤如下。



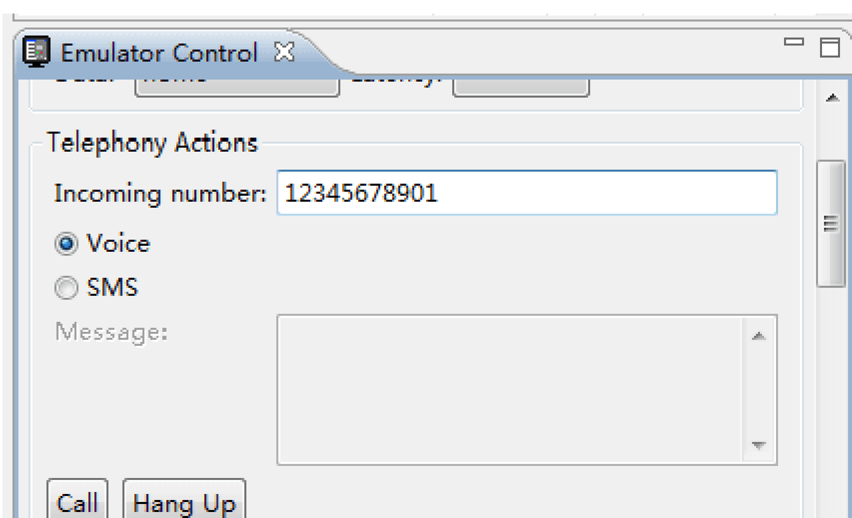


图 8-5　使用 DDMS 拨打电话



图 8-6　模拟器接听电话

（1）在 DDMS 中，选择需要接收 SMS 的模拟器。

（2）切换到模拟器控制选项卡。

（3）输入模拟发送的电话号码，可以包括任意数字、+和#。

（4）选中 SMS 单选按钮。

（5）输入 SMS 消息的正文。

（6）单击 Send 按钮。

（7）模拟器将会接收到 SMS 并显示通知。

操作过程如图 8-7 和图 8-8 所示。

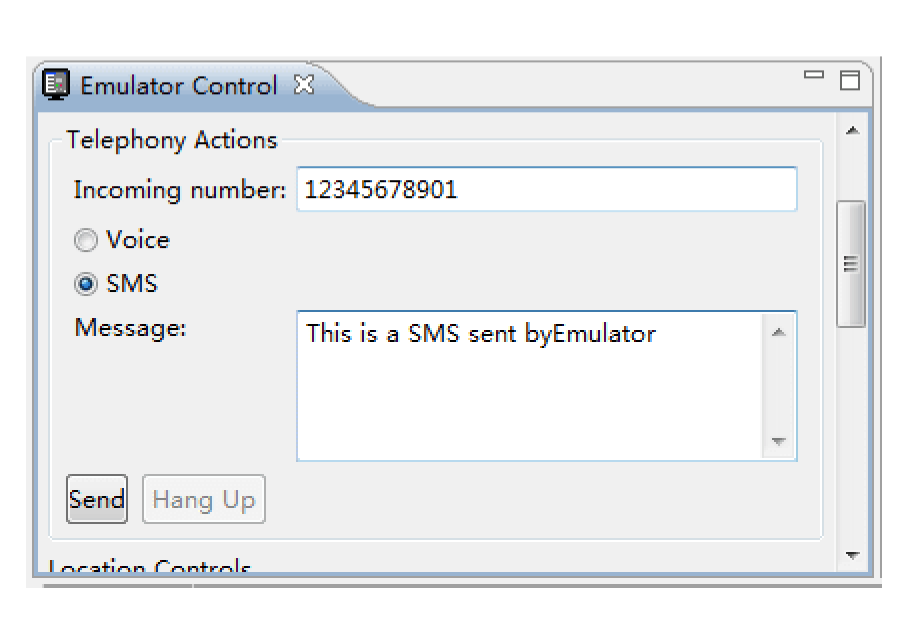


图 8-7　使用 DDMS 发送 SMS

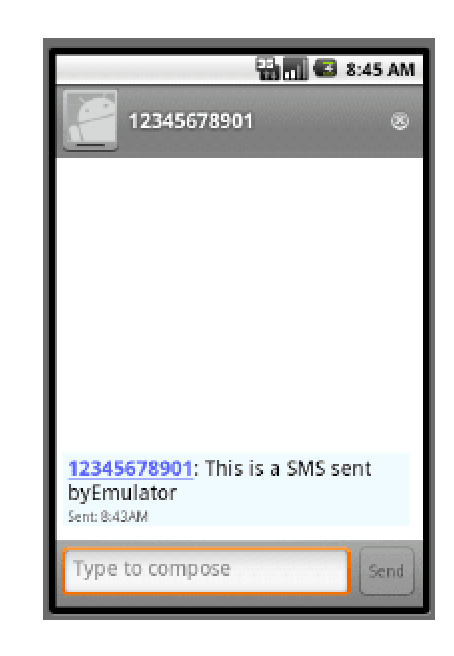


图 8-8　模拟器接收短信

3. 发送位置坐标

向模拟器发送 GPS 坐标，只需要在模拟器控制选项卡中简单地输入 GPS 坐标，单击 Send 按钮，就可以使用模拟器上的 Maps 应用程序接收当前位置了，如图 8-9 所示。

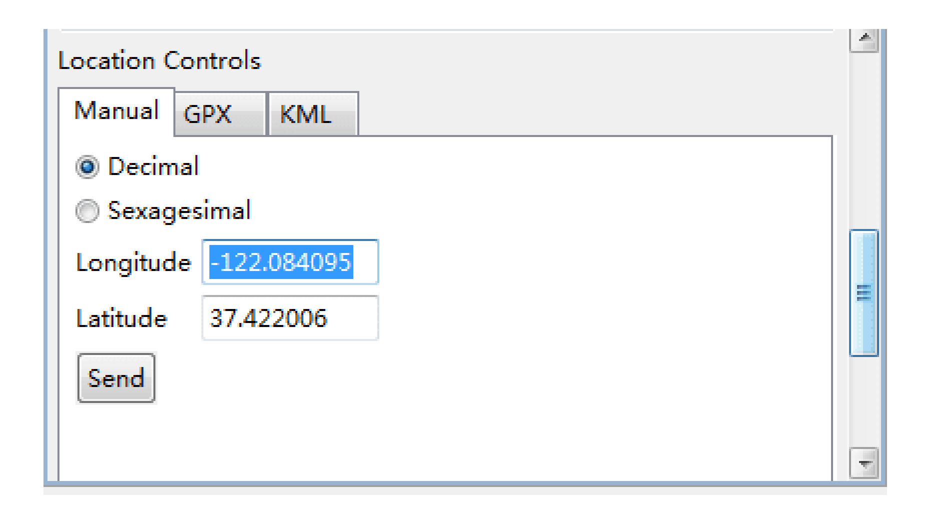
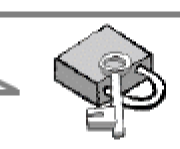
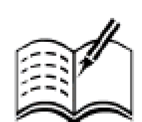


图 8-9　使用 DDMS 发送位置坐标




Note

8.6　使用程序日志 LogCat

DDMS 中融合了 LogCat 工具，LogCat 为 DDMS 用户界面底部的一个选项卡，可以通过单击内含字母的圆圈图标 Ⓥ Ⓓ Ⓘ Ⓦ Ⓔ 来控制信息的显示量。默认的Ⓥ代表 Verbose（即显示所有信息），其余可选图标包括Ⓓ（Debug，调试）、Ⓘ（Information，信息）、Ⓦ（Warning，警告）和Ⓔ（Error，错误）。

1. 增加 LogCat 视图

除了上面几种视图之外，LogCat 还可以创建自定义过滤标签以显示仅与调试标记（Debug Tag）相关的 LogCat 信息。可以通过+按钮来添加一个过滤标签以显示仅与特定标记匹配的日志信息。这对应用程序创建专有的调试标记将非常有用，这样，就可以过滤 LogCat，以保证只显示与应用程序相关的日志活动。

下面介绍在 Eclipse 中增加 LogCat 视图的方法。将过滤器命名为 Sysout 并且设置标记为 System.out。这样，就拥有了一个名为 Sysout 的 LogCat 标签，它将只显示 System.out 输出的日志信息。步骤如下：

（1）选择 Window/Show View/Other 菜单命令，在弹出的对话框中选择 Android/LogCat，如图 8-10 所示。

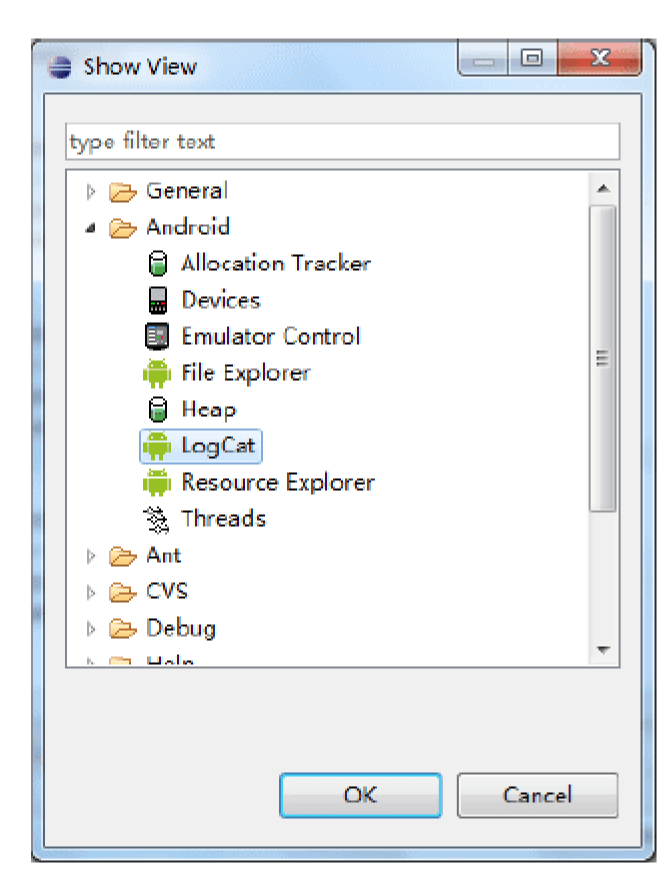


图 8-10　Show View 对话框

（2）单击 OK 按钮，在 Eclipse 中会增加 LogCat 视图，如图 8-11 所示。

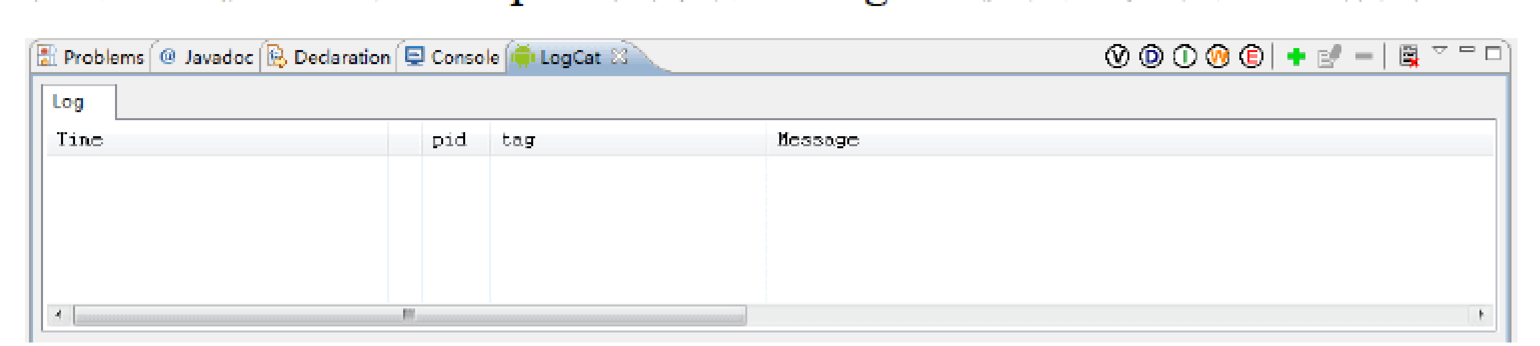
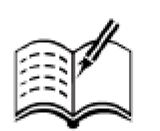


图 8-11　LogCat 视图


Note

（3）虽然有了 LogCat 视图，但是并不会显示 System.out.println()函数的输出信息，需要再增加一个过滤器，单击 ✚ 按钮，打开 Log Filter 对话框，如图 8-12 所示。

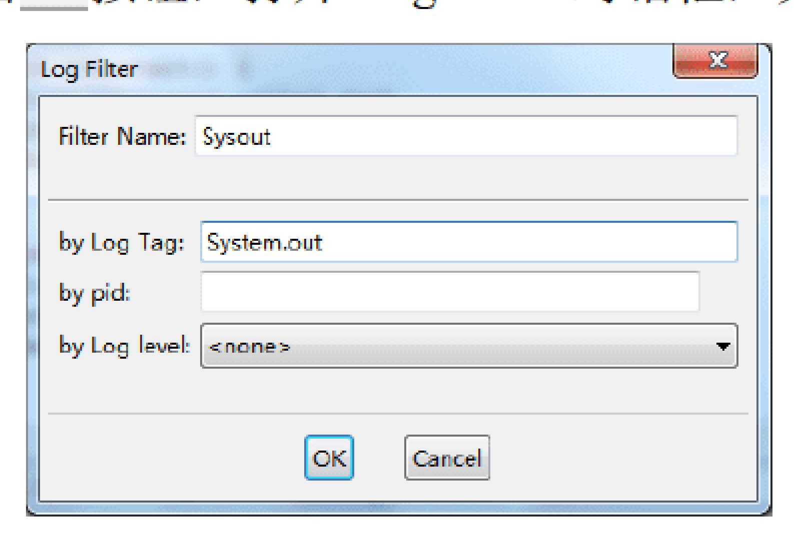


图 8-12　Log Filter 对话框

（4）单击 OK 按钮，就会增加一个 Sysout 过滤器，用来显示 System.out()函数输出的信息，如图 8-13 所示。

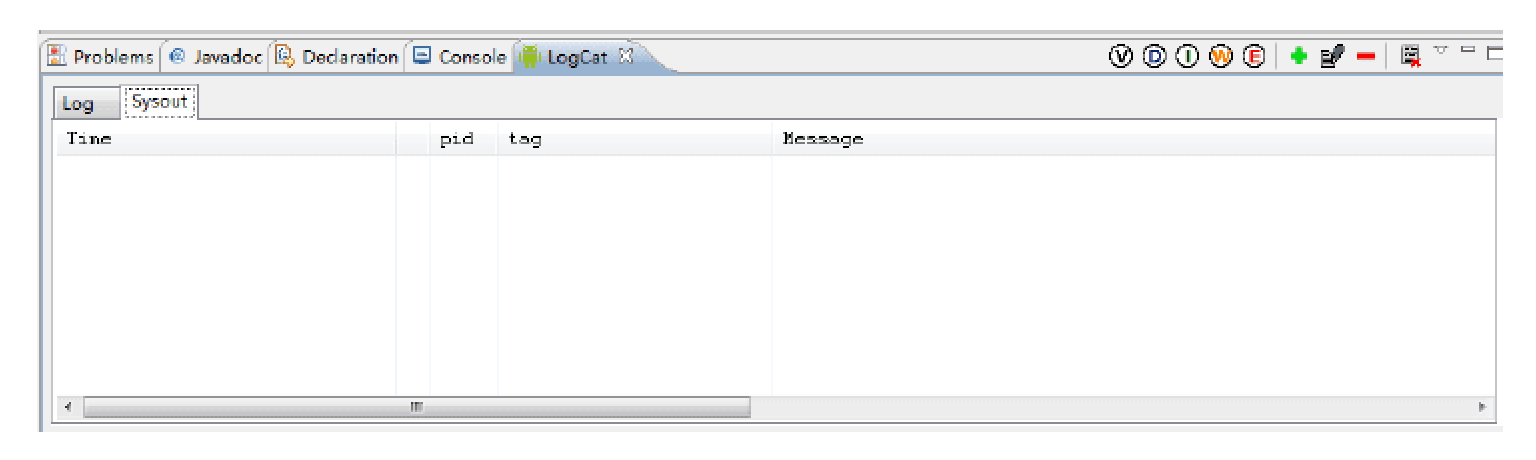


图 8-13　Sysout 过滤器

2. 通过 LogCat 获取错误信息

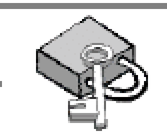
在进行 Android 程序的设计开发过程中，开发人员会遇到各种各样的错误。除了基本的语法错误之外，还有程序运行过程中发生的错误。对于语法错误，开发人员能够快速地找到，并根据提示进行修改。但是运行时产生的错误，就很难寻找原因，除了进行必要的异常处理外，更重要的是能够寻找到产生错误的原因，而 LogCat 就是获取此类错误信息的一个有效工具。

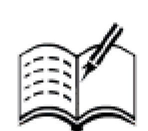
下面以第 7 章的 EX07_5 为例来说明如何通过 LogCat 获取错误。

在该项目中，从 AndroidManifest 中删除权限配置的设置，即删除<uses-permission android:name="android.permission.VIBRATE"/>。运行该程序，会产生一个错误，导致程序的退出。

当遇到此类错误时，仅仅根据程序的提示是无法知道程序的错误发生在什么地方的。但是程序的运行会在 LogCat 中形成日志，即程序的运行过程。当遇到此类错误时，可以通过查看 LogCat 获取发生错误的原因。在 LogCat 中，单击 Ⓔ 按钮，即可看到程序运行过程中所产生的错误及原因，如图 8-14 所示。

在图 8-14 中，可以看到发生的错误很多，那么究竟哪个才是发生错误的主要原因呢？在图中的下半部分（即图中选中行以下的部分）为异常堆栈的追踪信息，上半部分是产生异常的原因。在本例中，产生异常的原因是 java.lang.SecurityException: Requires VIBRATE permission。该错误提示已经明确告诉了产生错误的原因是需要 VIBRATE 权限，只要在




Note

AndroidManifest 中配置好该权限即可。因为在进行 Notification 消息提示时，使用的是 DEFAULT_ALL ，即使用所有默认值，需要有设置权限。

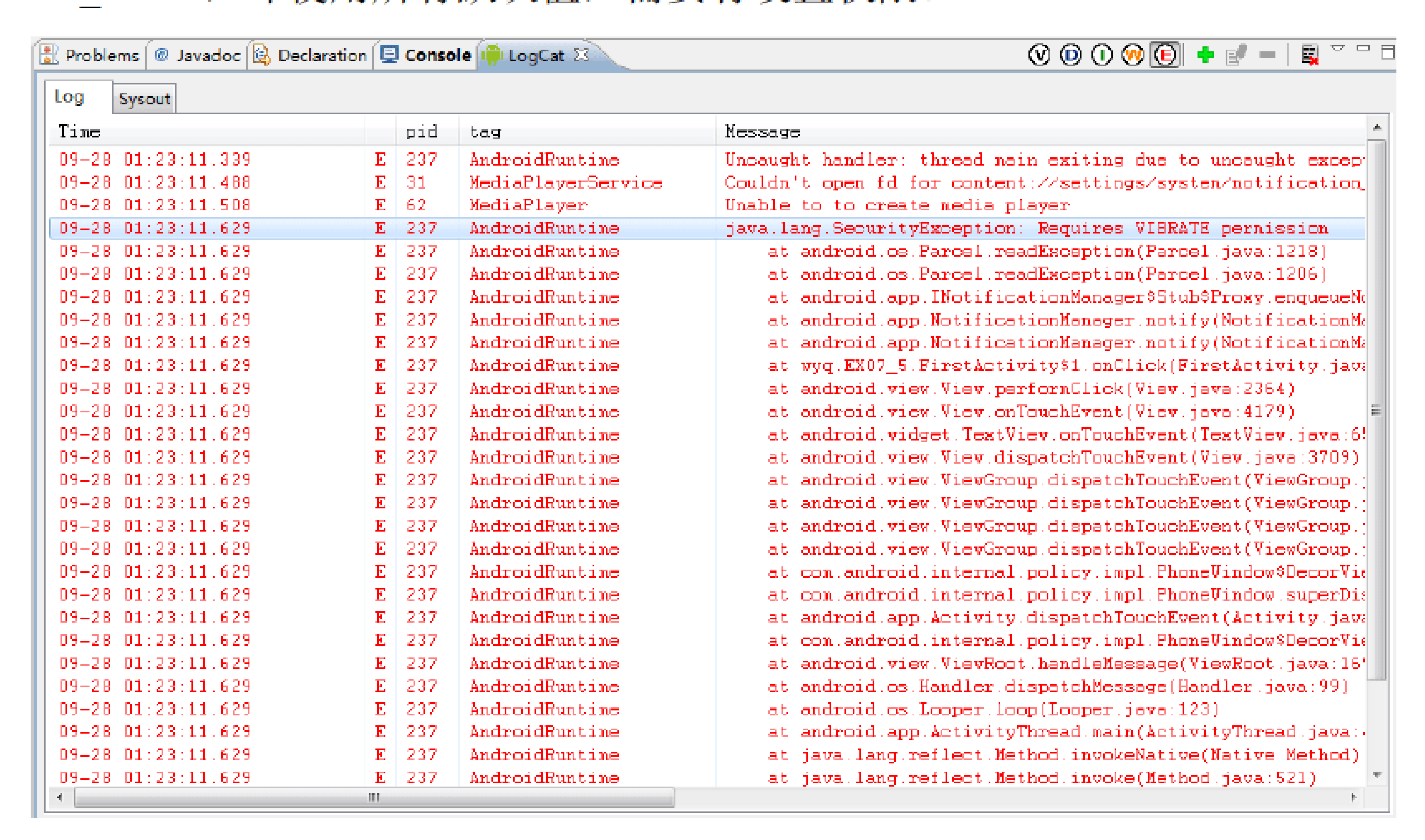


图 8-14　LogCat 错误信息

8.7　在模拟器或者目标设备上截屏

可以在 DDMS 中截取模拟器或设备的屏幕显示。设备屏幕对于调试来讲非常有用，它使 DDMS 工具特别适合 QA 人员，并且受到开发人员的欢迎。要进行屏幕截取，可以执行以下步骤。

（1）在 DDMS 中，选择需要截屏的模拟器或设备。

（2）在模拟器或设备上，确认屏幕显示的为想要截取的画面。

（3）单击带有方形彩色图案的图标进行截屏，此时将启动一个截屏窗口。

（4）在截屏窗口中，单击 Save 按钮保存屏幕截图。

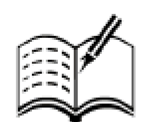
8.8　使用手机调试 Android 程序

Android 开发平台的模拟器运行速度非常慢，如果程序设计人员忍受不了其运行速度，可以在真实的手机上进行程序的调试。步骤如下：

（1）设置 Android 手机为 USB 调试模式，依次选择 menu/“设置”/“应用程序”/“开发”/“USB 调试”，如图 8-15 所示。

（2）用 USB 连接手机和计算机，在这一步需要安装手机的驱动程序。

（3）确定连接成功后，在命令行下进入 android SDK tools 所在的文件目录，输入命令：adb devices，如果在 List of devices attached 下面出现****** device 的字样，说明连接成功，其中******就是检测至的手机设备，如图 8-16 所示。


Note

（4）设置应用程序为调试模式（此操作为想要手机调试程序时使用，不设置也可以在手机上运行程序）。编辑 AndroidManifest.xml，增加调试参数 android:debuggable="true"，如下所示：

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
android:debuggable="true">
```



图 8-15　设置 USB 调试

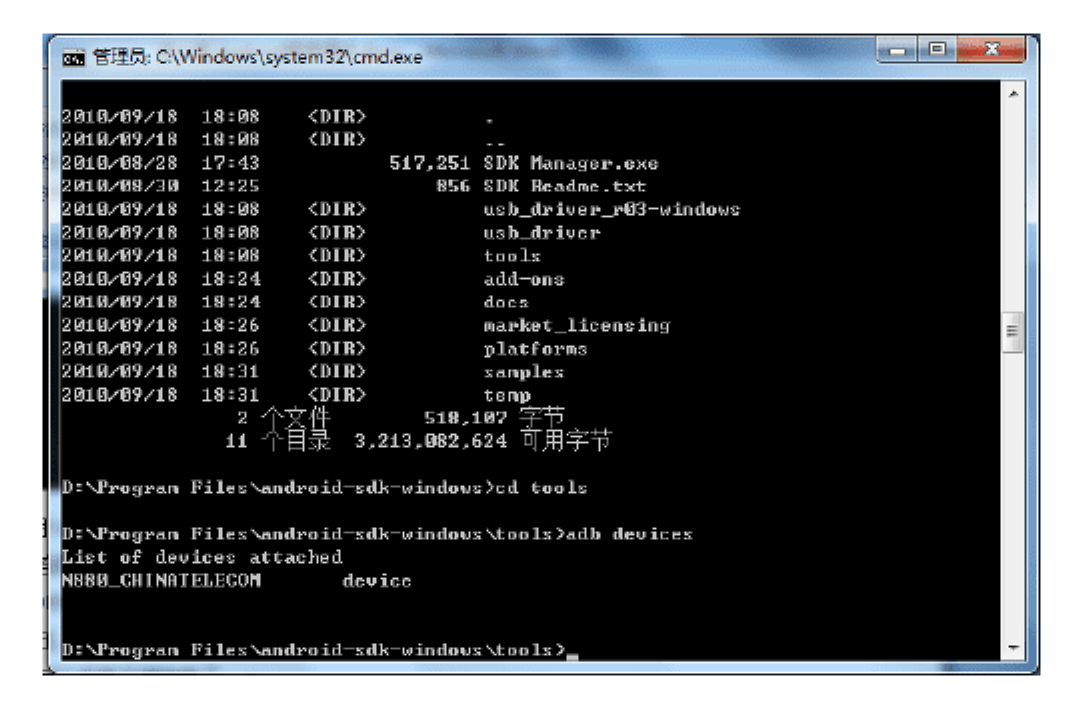


图 8-16　显示设备列表

（5）执行真机调试操作。在 Eclipse 调试对话框的 Target 选项卡中选中 Manual，单击 Debug 按钮，选择真机设备，开始调试，可以在 DDMS 中看到手机信息，并可以对手机进行截图，如图 8-17 所示。

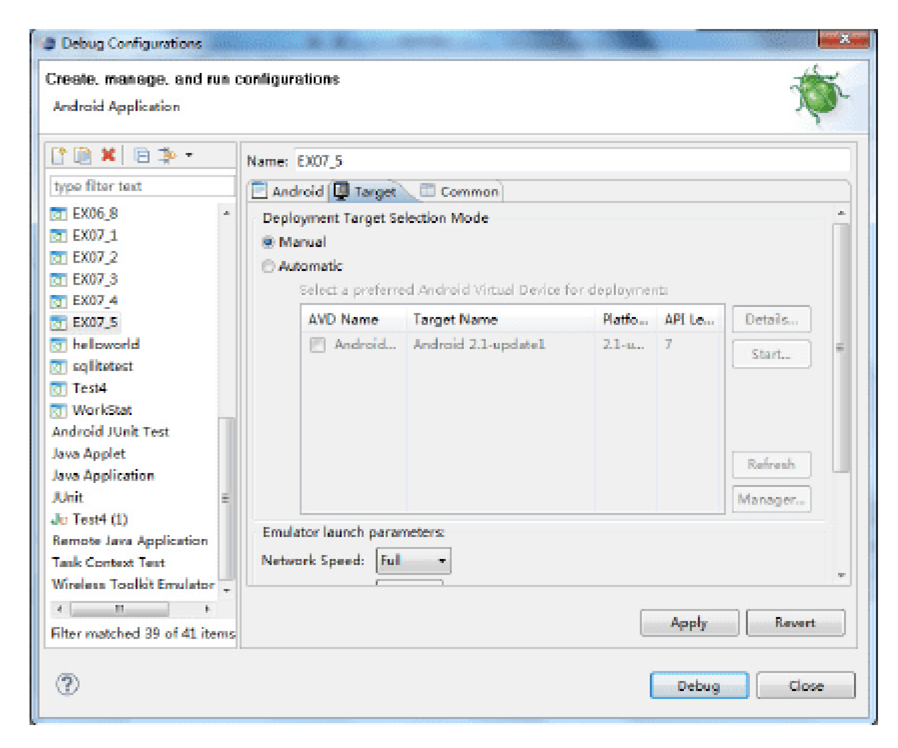




图 8-17　真机调试程序

8.9　习　　题

1．简述 DDMS 的运行原理。
2．通过 File Explorer 向模拟器中导入/导出文件。
3．将一个 Android 应用程序安装到手机上并且运行。
4．通过 LogCat 查看运行程序所产生的日志。
5．在 LogCat 中增加一个名为 DebugError 的过滤器，用于显示调试过程中的错误信息。

第 9 章

Android 数据存储与处理

【本章内容】

- ❑ 首选项
- ❑ 文件
- ❑ 数据库
- ❑ ContentProvider 类

无论是在桌面平台还是移动平台，应用程序都需要持久存储其数据，所以每个平台都提供了相应的数据存储机制。例如，Windows 平台提供了文件系统用于持久存储用户数据；J2ME 平台提供了记录管理系统（Record Management System，RMS）机制来存储用户的记录数据。在 Android 平台，主要提供了 3 种数据存储方式：首选项、文件和数据库。本章将分别介绍首选项、文件和数据库的使用方法。此外，Android 还提供了 ContentProvider 类来实现不同应用程序之间共享数据。

9.1 首 选 项

首选项（SharedPreferences）是一种轻量级的、用于存储或获取简单数据类型的“键-值”项的机制，可以将其想象为 Web 开发的 Cookies。它可以用键值对的方式把简单数据类型（boolean、int、float、long 和 string）存储在应用程序的私有目录下（data\data\[包名]\shared_prefs\）自己定义的 XML 文件中。其典型的用法是存储应用程序的首选项，如程序的基本设置等，这些选项将在应用程序启动时被载入，大多数应用程序都提供了首选项设置的功能。

9.1.1 SharedPreferences 类简介

SharedPreferences 保存的数据主要是类似于配置信息格式的数据，因此保存的数据主要是简单类型的键值对（key-value），它保存的是一个 XML 文件，常用的方法及说明如表 9-1 所示。

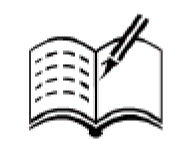

Note

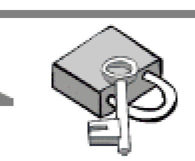
表 9-1 SharedPreferences 常用的方法及说明

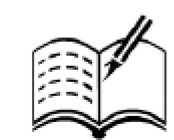
方 法 名 称	参 数 说 明	描 述
public abstract boolean contains (String key)	key：想要判断的 preference 的名称	判断 preferences 是否包含一个 preference。如果 preferences 中存在 preference，则返回 true，否则返回 false
public abstract SharedPreferences.Editor edit ()		针对 preferences 创建一个新的 Editor 对象，通过它可以修改 preferences 里的数据，并且原子化地将这些数据提交回 SharedPreferences 对象。返回一个 SharedPreferences.Editor 的新实例，允许用户修改 SharedPreferences 对象里的值
public abstract Map<String, ?> getAll ()		取得 preferences 里面的所有值。返回一个 map，其中包含一列 preferences 中的键值对
public abstract XXX get XXX (String key, XXX defValue)	key：获取的 preference 的名称 defValue：当此 preference 不存在时返回的默认值	从 preferences 中获取一个 XXX 类型的值。其中 XXX 可以是 boolean、float、int、long、string 等基本数据类型。如果 preference 存在，则返回 preference 的值，否则返回 defValue
public abstract void registerOnSharedPreferenceChangeListener (SharedPreferences. OnSharedPreferenceChangeListener listener)	listener：将会被调用的回调函数	注册一个回调函数，当一个 preference 发生变化时调用
public abstract void unregisterOnSharedPreferenceChangeListener (SharedPreferences.OnSharedPreferenceChangeListener listener)	listener：要被注销的回调函数	注销一个之前（注册）的回调函数

SharedPreferences 是一个接口，而且在这个接口里并没有提供写入和读取数据的能力。但是在其内部有一个 Editor 内部的接口，该接口有一系列的方法用于操作 SharedPreferences。Editor 接口的常用方法如表 9-2 所示。

表 9-2 Editor 接口的常用方法

方 法 名 称	描 述
Public abstract SharedPreferences.Editor clear()	清空 SharedPrefereces 里所有的数据
Public abstract Boolean commit()	当 Editor 编辑完成后，调用该方法可以提交修改，而且必须要调用该数据才修改
SharedPreferences.Editor remove(String key)	删除 SharedPreferences 里指定 key 对应的数据项
SharedPreferences.Editor putXXX(String key, XXX value)	向 SharedPreferences 中存入指定的 key 对应的数据，其中 XXX 可以是 boolean、float、int、long、string 等基本数据类型





SharedPreferences 只是一个接口，程序是无法创建 SharedPreferences 实例的，可以通过 Context.getSharedPreferences(String name,int mode)来得到一个 SharedPreferences 实例，其参数介绍如下。

- name：是指文件名称，不需要加扩展名.xml，系统会自动添加上文件扩展名。一般该文件存储在\data\data\<package name>\shared_prefs 下。
- mode：是指定读写方式，其值有 3 种，分别如下。
  - Context.MODE_PRIVATE：指定该 SharedPreferences 数据只能被本应用程序读、写。
  - Context.MODE_WORLD_READABLE：指定该 SharedPreferences 数据能被其他应用程序读，但不能写。
  - Context.MODE_WORLD_WRITEABLE：指定该 SharedPreferences 数据能被其他应用程序读写。

SharedPreferences 的使用步骤如下：

首先，创建首选项 XML 文件来描述首选项，在 res\xml\目录下的 XML 文件中定义首选项。Android 提供 Preference 这个键-值对的方式来处理这种情况，自动保存这些数据，并立刻生效，同时 Android 提供一种类似的 layout 方式来进行 Preference 的布局。

Preference 的组织方式有 PreferenceScreen 和 PreferenceCategory，PreferenceCategory 是带层次组织关系，而 PreferenceScreen 就是最基础的方式。

在 XML 文件中定义了一个 PreferenceScreen，然后创建 ListPreference 作为子屏幕。对于 PreferenceScreen，设置了 3 个属性：key、title 和 summary。key 是一个字符串，可用于以编程的方式表示项（类似于使用 android:id 的方式）；title 表示标题；summary 表示用途。PreferenceScreen 的常用属性如表 9-3 所示。

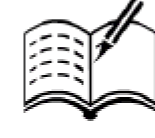
表 9-3　PreferenceScreen 的常用属性

属　性	说　明
android:key	选项的名称或键（如 selected_flight_sort_option）
android:title	选项的标题
android:summary	选项的简短摘要
android:entries	可将选项设置成列表项的文本
android:entryValues	定义每个列表项的值。注意：每个列表项有一些文本和一个值。文本由 entries 定义，值由 entryValues 定义
android:dialogTitle	对话框的标题，在视图显示为模态对话框时使用
android:defaultValue	项列表中选项的默认值

其次，要向用户显示首选项，编写一个活动类来扩展预定义的 Android 类 android.preference.PreferenceActivity，然后使用 addPreferencesFromResource()方法将资源添加到活动的资源集合中。

9.1.2　SharedPreferences 使用实例

本节将通过实例介绍 SharedPreferences 的使用方法。在本实例中，可以在主界面设置




账户，也可以通过选项菜单打开程序的设置账户及其他选项，然后在主界面显示程序设置的结果。

本实例的开发步骤如下：

（1）新建项目 EX09_1。

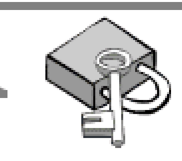
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

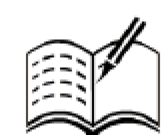


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8       android:layout_width="fill_parent"
9       android:layout_height="wrap_content"
10      android:text="这是一个首选项 SharedPreferences 示例"
11      />
12 <TextView
13      android:layout_width="fill_parent"
14      android:layout_height="wrap_content"
15      android:text="输入用户名："
16      />
17 <EditText
18      android:layout_width="fill_parent"
19      android:layout_height="wrap_content"
20      android:id="@+id/user"
21      />
22 <Button
23      android:layout_width="wrap_content"
24      android:layout_height="wrap_content"
25      android:id="@+id/bt_OK"
26      android:text="确定"
27      android:gravity="center_horizontal"
28      />
29 <TextView
30      android:layout_width="fill_parent"
31      android:layout_height="wrap_content"
32      android:text="首选项的设置为："
33      />
34 <TextView
35      android:layout_width="fill_parent"
36      android:layout_height="wrap_content"
37      android:id="@+id/tv"
38      />
39 </LinearLayout>
```

说明：

- 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。

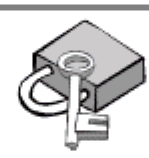




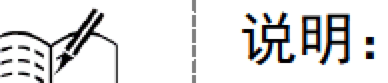
- ❑ 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- ❑ 第 12~16 行：定义一个 TextView 控件及其大小、文本。
- ❑ 第 17~21 行：定义一个 id 为 user 的 EditText 控件及其大小。
- ❑ 第 22~28 行：定义一个 id 为 bt_OK 的 Button 控件及其大小、对齐方式。
- ❑ 第 29~33 行：定义一个 TextView 控件及其大小、文本。
- ❑ 第 34~38 行：定义一个 id 为 tv 的 TextView 控件及其大小，用于显示信息。

（3）创建首选项 XML 文件来描述首选项，在 res\xml 文件夹下创建 ex091preference.xml 文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:key="setting"
 android:title="软件设置">
 <PreferenceCategory
      android:key="basicSet"
      android:title="基本设置">
          <EditTextPreference
               android:key="username"
               android:title="账户"
               android:defaultValue=" "
               android:summary="设置账户名"
               />
           <CheckBoxPreference
               android:key="nightmode"
               android:title="夜间模式"
               android:summaryOn="已启用"
               android:summaryOff="未启用"
               />
            <RingtonePreference
              android:key="ringtone"
              android:title="铃声"
              android:showSilent="true"
              android:ringtoneType="alarm"
              android:summary="设置通知铃声"
          />
</PreferenceCategory>
<PreferenceCategory
      android:key="textSet"
      android:title="文本设置">
           <ListPreference
               android:key="fontSize"
               android:title="字体大小"
               android:summary="设置字体大小"
               android:entries="@array/fontsize"
                android:entryValues="@array/fontsizevalue"
                android:dialogTitle="选择字体大小"
```



```
39                  />
40 </PreferenceCategory>
41 </PreferenceScreen>
```




说明：

- 第 2~5 行：定义一个 PreferenceScreen，其键为 setting，标题为“软件设置”。
- 第 6~8 行：定义一个 PreferenceCategory，其键为 basicSet，标题为“基本设置”，用于把下面的 3 个组件组织起来。
- 第 9~14 行：定义一个 EditTextPreference，其键为 username，标题为“账户”，摘要为“设置账户名”。当单击该组件时，弹出输入框进行输入。
- 第 15~20 行：定义一个 CheckBoxPreference，其键为 nightmode，标题为“夜间模式”。第 18 行设置当该复选框被选中时显示的摘要，第 19 行设置当该复选框未被选中时显示的摘要。
- 第 21~27 行：定义一个 RingtonePreference，其键为 ringtone，标题为“铃声”，摘要为设置通知铃声。
- 第 29~40 行：定义一个 PreferenceCategory，其键为 textSet，标题为“文本设置”，其中包含一个 ListPreference 组件。

  第 32~39 行定义一个 ListPreference，其键为 fontSize，标题为“字体大小”，摘要为“设置字体大小”。其中，第 36 行定义该列表显示的内容，该数组需要在 string.xml 资源文件中定义；第 37 行定义该列每项显示内容的值，该数组也需要在 string.xml 资源文件中定义；第 38 行设置所显示的对话框的标题。

（4）编写 string 资源文件，在 string.xml 文件中加入以下代码：

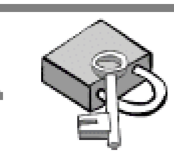
```
<string-array name="fontsize">
        <item>小</item>
        <item>正常</item>
        <item>大</item>
  </string-array>
  <string-array name="fontsizevalue">
        <item>0</item>
        <item>1</item>
        <item>2</item>
  </string-array>
```

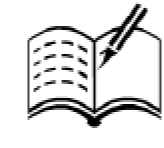
说明：

定义 fontsize 和 fontsizevalue 两个数组，分别用于首选项布局文件中 ListPreference 组件的显示项及对应项的值。

（5）创建首选项的活动类 SetPreferenceActivity，派生于 PreferenceActivity 类，编写代码如下：

```
package wyq.EX09_1;
import android.os.Bundle;
import android.preference.PreferenceActivity;
public class SetPreferenceActivity extends PreferenceActivity {
```





Note

```
5     @Override
6     protected void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         addPreferencesFromResource(R.xml.ex091preference);
9     }
10 }
```

说明：

第 8 行：调用 addPreferencesFromResource()为 SetPreferenceActivity 传入 R.xml.ex091 preference 资源。

（6）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

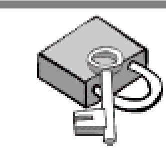
```
package wyq.EX09_1;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class FirstActivity extends Activity {
    private TextView tv;
    private EditText edtUser;
    private Button bt_OK;
    private SharedPreferences pref;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        edtUser=(EditText)findViewById(R.id.user);
        bt_OK=(Button)findViewById(R.id.bt_OK);
        bt_OK.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                pref= getSharedPreferences("wyq.EX09_1_preferences",
                    MODE_WORLD_WRITEABLE);
                Editor editor=pref.edit();
                String userName=edtUser.getText().toString();
                editor.putString("username", userName);
                editor.commit();
```

```
                    showSetting();
                }
            });
        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu)
        {
            menu.add(Menu.NONE, Menu.FIRST + 1, 1, "设置");
            menu.add(Menu.NONE, Menu.FIRST + 2, 2, "退出");
            return true;
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item)
        {
            switch (item.getItemId())
            {
                case Menu.FIRST + 1: Intent intent=new Intent();
                                intent.setClass(this, SetPreferenceActivity.class);
                                startActivityForResult(intent,1);
                                break;
                case Menu.FIRST + 2:finish();
                                break;
            }
            return super.onOptionsItemSelected(item);
        }
        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
             // TODO Auto-generated method stub
             super.onActivityResult(requestCode, resultCode, data);
             showSetting();
        }
        private void showSetting()
        {
            String settingStr;
            tv=(TextView)findViewById(R.id.tv);
            pref= getSharedPreferences("wyq.EX09_1_preferences",
                MODE_WORLD_READABLE);
            String username=pref.getString("username", "");
            settingStr="您设置的账户名为:"+username+"\n";

            Boolean nightmode=pref.getBoolean("nightmode",false);
            if(nightmode)
            {
                settingStr=settingStr+"夜间模式：已启用\n";
            }
            else
            {
                settingStr=settingStr+"夜间模式：未启用\n";
            }
```



```
85          String fontSize = pref.getString("fontSize", "0");
86          if(fontSize.equals("0"))
87          {
88              settingStr=settingStr+"字体：小"+"\n";
89          }
90          if(fontSize.equals("1"))
91          {
92              settingStr=settingStr+"字体：正常"+"\n";
93          }
94          else if(fontSize.equals("2"))
95          {
96              settingStr=settingStr+"字体：大"+"\n";
97          }
98          tv.setText(settingStr);
99      }
100 }
```

说明：

- 第 16~19 行：定义 TextView、EditText、Button、SharedPreferences 对象。
- 第 25 行：获取 EditText 控件的引用。
- 第 26 行：获取 Button 控件的引用。
- 第 27~40 行：为 Button 控件添加单击监听事件。
  - 第 32 行：获取当前的首选项文件，第一个参数用来指定存储首选项值的文件的名称，格式为“包名_preferences”，本项目的包名为 wyq.EX09_1，所以文件名为 wyq.EX09_1_preferences；第二个参数为打开模式。在本 Activity，要将 EditText 中输入的账户保存在首选项文件中，所以打开模式为 MODE_WORLD_WRITEABLE。
  - 第 33 行：获取首选项的编辑器。
  - 第 34 行：获取文本框的内容。
  - 第 35 行：将文本框的内容写入到首选项中，第一个参数为要写入的首选项的键；第二个参数为该键的值，即从文本框中获取到的内容。
  - 第 36 行：提交修改。在编辑首选项后，一定要进行提交，否则不会写入到首选项文件中。
  - 第 37 行：调用 showSetting()用于在 TextView 中显示首选项的设置。
- 第 41~47 行：创建选项菜单。
- 第 48~61 行：为选项菜单增加事件。其中，第 55 行，单击选项菜单中的“设置”时，程序跳转到首选项设置的 Activity，并且因为设置完成后要显示设置的结果，所以使用 startActivityForResult()，而不是 startActivity()。
- 第 62~67 行：重写 onActivityResult()函数，再调用 showSetting()函数，显示首选项设置的结果。
- 第 68~100 行：定义 showSetting()函数。
  - 第 72 行：获取当前的首选项文件，打开模式为 MODE_WORLD_READABLE。

- 第 73 行：获取首选项中的 username 键的值，第一个参数为键名，第二个参数为默认值。
- 第 76 行：获取首选项中的 nightmode 键的值。对于复选框，其选中或者未被选中使用 boolean 来表示，所以使用 getBoolean()函数来获取复选框首选项组件的值。
- 第 85 行：获取列表首选项组件的值，获取到的是每一项对应的 value 及 fontsizevalue 数组中的值，而不是显示的值。
- 第 98 行：设置 TextView 的显示内容。在获取到各个首选项组件的值后，拼接了 settingStr 字符串，用于描述首选项的设置内容。

本实例运行结果如图 9-1~图 9-3 所示。



图 9-1　EX09_1 运行结果



图 9-2　设置首选项



图 9-3　显示首选项

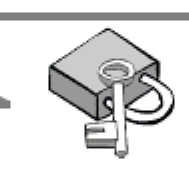
9.2　文　　件

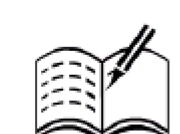
和桌面平台一样，Android 平台允许应用程序在移动设备或者移动存储设备上直接存储文件。不同的是，某一应用程序所存储的文件是不能被其他应用程序访问的。Android 的文件系统是基于 Linux 并且支持基于模式的权限，访问该文件系统的方式有很多。在应用程序中可以创建和读取文件、访问作为资源使用的原始文件，还可以创建和访问自定义 XML 文件。

9.2.1　文件访问

1．创建文件

在 Android 平台中，可以轻松地创建文件，并将创建的文件存储在文件系统中当前应用程序的数据路径下。Android 提供了一种简单的方法来创建文件，用 OpenFileOutput 获取 FileOutputStream 引用，来获取创建文件的数据流，该文件将存储在 Android 平台下的“data\data\[包名]\files\文件名”。创建数据流之后，可以使用传统的 Java 访问方法向其写入数据。在文件创建后，可以通过 adb（Android Debug Bridge）工具或者 DDMS 的 File Explorer 从 Android 平台获取文件。





Note

2．访问文件

访问文件与创建文件是两个相反的操作。在输入时，可以使用 OpenFileInput 获取 FileInputStream 引用，来获取读取文件的数据流，然后通过 Java 方法来读取文件。

3．访问资源文件

资源文件存放在 res\raw 中。如果系统在应用程序中需要使用任何形式（文本、图像、文档）的原始文件，则可以将该文件放在 res\raw 下。存储在 res\raw 的文件不会被平台编译，而是作为可用的原始资源。获取原始资源文件与获取文件类似，获取一个 InputStream，然后将该数据流分配给一个原始资源的引用。通过 getResources()获取 Resource 的引用，然后调用 openRawResource (int id)链接到特定的资源上，该 ID 可以从 R.raw.ID 获取到。

4．XML 文件

XML 用于标记电子文件，使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。在 Android 平台中，自定义的 XML 文件存储在 res\xml 中。XML 文件在程序部署时将被编译为有效的二进制类型。要处理 XML 资源，需要使用 XmlPullParser 类，该类使用 SAX 解析器遍历 XML。该解析器为遇到的每个元素都提供了一个由整型数据表示的事件类型，如 START_TAG、END_TAG、START_DOCUMENT、END_DOCUMENT。使用 next()方法来检索当前的时间类型值并将它与类中的事件进行比较。遇到每个元素都有一个名称、文本值和可选的属性，通过 getAttributeCount()获取每个项目的属性数，并通过 getAttributeName()与 getAttributeValue()获取节点的名称和值。

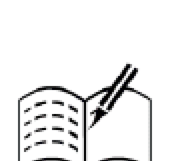
9.2.2　文件访问实例

本节将通过实例演示各种文件的访问方式。本实例的开发步骤如下：

（1）创建项目 EX09_2。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
>
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="这是一个文件操作的示例"
/>
<Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```


Note

```
15        android:id="@+id/bt_createFile"
16        android:text="创建文件"
17 />
18 <Button
19        android:layout_width="fill_parent"
20        android:layout_height="wrap_content"
21        android:id="@+id/bt_readFile"
22        android:text="读取文件"
23 />
24 <Button
25        android:layout_width="fill_parent"
26        android:layout_height="wrap_content"
27        android:id="@+id/bt_readRawFile"
28        android:text="读取资源文件"
29 />
30 <Button
31        android:layout_width="fill_parent"
32        android:layout_height="wrap_content"
33        android:id="@+id/bt_readXMLFile"
34        android:text="读取 XML 文件"
35 />
36 <Button
37        android:layout_width="fill_parent"
38        android:layout_height="wrap_content"
39        android:id="@+id/bt_createXMLFile"
40        android:text="创建 xml 文件"
41 />
42 </LinearLayout>
```

说明：

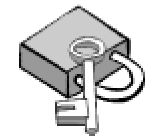
- ❑ 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。
- ❑ 第 7~11 行：定义一个 TextView 控件及其大小、文本。
- ❑ 第 12~17、18~23、24~29、30~35、36~41 行：依次定义 Button 控件及其大小、文本、ID。

（3）修改主 Activity 的类文件 FirstActivity.java，编写代码如下：

```
package wyq.EX09_2;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

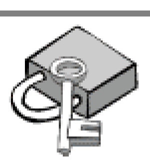
public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private Button bt_createFile;
    private Button bt_readFile;
```



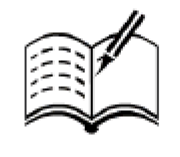



```
private Button bt_readRawFile;
private Button bt_readXMLFile;
private Button bt_createXMLFile;
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  bt_createFile=(Button)findViewById(R.id.bt_createFile);
  bt_readFile=(Button)findViewById(R.id.bt_readFile);
  bt_readRawFile=(Button)findViewById(R.id.bt_readRawFile);
  bt_readXMLFile=(Button)findViewById(R.id.bt_readXMLFile);
  bt_createXMLFile=(Button)findViewById(R.id.bt_createXMLFile);

      bt_createFile.setOnClickListener(new btClickListener());
      bt_readFile.setOnClickListener(new btClickListener());
      bt_readRawFile.setOnClickListener(new btClickListener());
      bt_readXMLFile.setOnClickListener(new btClickListener());
      bt_createXMLFile.setOnClickListener(new btClickListener());
}
class btClickListener implements View.OnClickListener
{
      Intent intent=new Intent();
      @Override
      public void onClick(View v) {
            // TODO Auto-generated method stub
            switch (v.getId())
            {
            case R.id.bt_createFile:
                        intent.setClass(FirstActivity.this, CreateFileActivity.class);
                        startActivity(intent);
                        break;
            case R.id.bt_readFile:
                        intent.setClass(FirstActivity.this, ReadFileActivity.class);
                        startActivity(intent);
                        break;
            case R.id.bt_readRawFile:
                        intent.setClass(FirstActivity.this, ReadRawFileActivity.class);
                        startActivity(intent);
                        break;
            case R.id.bt_readXMLFile:
                        intent.setClass(FirstActivity.this, ReadXmlFileActivity.class);
                        startActivity(intent);
                        break;
            case R.id.bt_createXMLFile:
                        intent.setClass(FirstActivity.this, CreateXmlFileActivity.class);
                        startActivity(intent);
                        break;
            }
      }
```



```
62     }
63 }
```



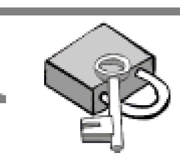
Note

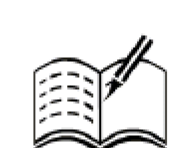
说明：

- 第 11~15 行：定义 5 个 Button 类对象。
- 第 20~24 行：获取 Button 控件的引用。
- 第 26~30 行：为每个 Button 控件添加单击监听事件。参数是一个 btClickListener 类对象，具体的监听事件在 btClickListener 类中实现。
- 第 32~62 行：实现 btClickListener 类和 OnClickListener 接口。在 btClickListener 类中，根据所单击 Button 的 ID 的不同，进行不同的事件处理。在这里就是单击不同的 Button，跳转到不同的 Activity 中。第 38 行 v.getId()用于获取所单击的 Button 的 ID。

（4）创建 createfile.xml 文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
  <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入文件内容:"
        />
   <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:scrollHorizontally="true"
        android:id="@+id/edt_file"
        android:lines="10"
        />
   <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入文件名:"
        />
   <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edt_filename"
        />
   <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/bt_saveFile"
        android:text="保存文件"
```






```
      />
</LinearLayout>
```

说明：

在本布局文件中，先定义一个纵向的线性布局。在线性布局中定义了两个 TextView、两个 EditText 与一个 Button 控件。第一个 EditText 控件用于输入要保存的文件内容，第二个 EditText 控件用于输入文件名，Button 控件作为保存命令按钮产生保存事件。

（5）创建 CreateFileActivity 类文件 CreateFileActivity.java，在这个类中，输入文件内容与文件名后，单击“保存”按钮保存文件。编写代码如下：

```
package wyq.EX09_2;

import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CreateFileActivity extends Activity {
    private EditText edt_file;
    private EditText edt_filename;
    private Button bt_saveFile;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.createfile);
        edt_file=(EditText)findViewById(R.id.edt_file);
        edt_filename=(EditText)findViewById(R.id.edt_filename);
        bt_saveFile=(Button)findViewById(R.id.bt_saveFile);

        bt_saveFile.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                FileOutputStream fos=null;
                String filename=edt_filename.getText().toString();
                try
                {
                    fos=openFileOutput(filename,Context.MODE_PRIVATE);
                    fos.write(edt_file.getText().toString().getBytes());
                }
                catch(Exception e)
```

Note

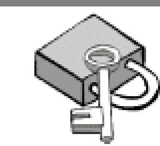
```
39                {
40                    Toast.makeText(CreateFileActivity.this, "文件保存失败",
                      Toast.LENGTH_LONG).show();
41                }
42                finally
43                {
44                    if(fos!=null)
45                    {
46                        try
47                        {
48                            fos.flush();
49                            fos.close();
50                        }
51                        catch(IOException e)
52                        {}
53                    }
54                }
55            }
56        });
57    }
58 }
```

说明：

- 第 14~16 行：定义两个 EditText 控件对象与一个 Button 控件对象。
- 第 22~24 行：获取控件的引用。
- 第 26 行：为 Button 控件添加单击监听事件，实现文件的保存。
- 第 31 行：定义一个 FileOutputStream 文件输出流的对象。
- 第 33~37 行：将文件的操作放入一个 try…catch 中，获取文件操作的异常。
  - 第 35 行：打开文件，用 openFileOutput()获取 FileOutputStream 引用，获取创建文件的数据流。第一个参数为创建的文件的文件名，第二个参数为打开模式。
  - 第 36 行：写入文件内容。
- 第 40 行：当文件操作出现异常时，使用 Toast 显示错误提示信息。
- 第 48、49 行：当文件操作完毕后，关闭文件。

（6）创建 readfile.xml 文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="输入文件名:"
```



```
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edt_fname"
        />
    <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/bt_readOutFile"
          android:text="读取文件"
          />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/tv_filecontent"
        />
</LinearLayout>
```

说明：

在本布局文件中，先定义一个纵向的线性布局。在线性布局中依次定义两个 TextView 控件，第二个 TextView 用于显示文件的内容。

（7）创建 ReadFileActivity 类文件 ReadFileActivity.java，在这个类中，输入文件名后，单击按钮显示所读取文件的内容。编写代码如下：

```
package wyq.EX09_2;

import java.io.FileInputStream;
import java.io.IOException;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

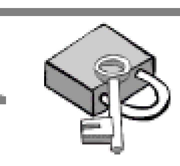
public class ReadFileActivity extends Activity {
    private EditText edt_fname;
    private TextView tv_fileContent;
    private Button bt_readoutfile;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.readfile);

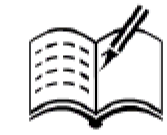
        edt_fname=(EditText)findViewById(R.id.edt_fname);
```

```
24          tv_fileContent=(TextView)findViewById(R.id.tv_filecontent);
25          bt_readoutfile=(Button)findViewById(R.id.bt_readOutFile);
26
27          bt_readoutfile.setOnClickListener(new Button.OnClickListener()
28          {
29              @Override
30              public void onClick(View v) {
31                  // TODO Auto-generated method stub
32                  FileInputStream fis=null;
33                  String filename=edt_fname.getText().toString();
34                  try
35                  {
36                      fis=openFileInput(filename);
37                      byte[] reader=new byte[fis.available()];
38                      while(fis.read(reader)!=-1)
39                      {}
40                      tv_fileContent.setText(new String(reader));
41                  }
42                  catch(Exception e)
43                  {
44                      Toast.makeText(ReadFileActivity.this, "文件读取失败",
                        Toast.LENGTH_LONG).show();
45                  }
46                  finally
47                  {
48                      if(fis!=null)
49                      {
50                          try
51                          {
52                              fis.close();
53                          }
54                          catch(IOException e)
55                          {
56                          }
57                      }
58                  }
59              }
60          });
61      }
62 }
```

说明：

- 第 14~16 行：分别定义 EditText、TextView、Button 控件对象。
- 第 23~25 行：获取控件的引用。
- 第 27 行：为 Button 控件添加单击监听事件，实现文件的读取。
- 第 32 行：定义一个 FileInputStream 文件输入流的对象。
- 第 36 行：使用 openFileInput()获取 FileInputStream 引用，来获取读取文件的数据流。
- 第 37 行：根据文件输入流的大小定义一个二进制数据数组，存放读取出的文件





内容。

- 第 40 行：在 TextView 控件中显示文件内容。
- 第 44 行：当文件操作出现异常时，使用 Toast 显示错误提示信息。
- 第 52 行：当文件操作完毕后，关闭文件。

（8）创建 readrawfile.xml 文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="读取 res/raw/wang.txt 文件:"
        />
     <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/tv_rawfilecontent"
        />
</LinearLayout>
```

说明：

在本布局文件中，先定义一个纵向的线性布局。在线性布局中依次定义两个 TextView 控件，第二个 TextView 用于显示资源文件的内容。

（9）创建 ReadRawFileActivity 类文件 ReadRawFileActivity.java。首先在 res 下新建 raw 文件夹，用于存放原始资源文件，然后向该文件夹复制 wang.txt 文件。在这个类中，将读取 wang.txt，显示该文件的内容。编写代码如下：

```
package wyq.EX09_2;

import java.io.IOException;
import java.io.InputStream;
import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

public class ReadRawFileActivity extends Activity {
    private TextView tv_filerawContent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
```

Note

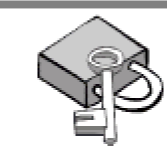
```
17         setContentView(R.layout.readrawfile);
18
19         tv_filerawContent=(TextView)findViewById(R.id.tv_rawfilecontent);
20         Resources rs=this.getResources();
21         InputStream is=null;
22         try
23         {
24             is=rs.openRawResource(R.raw.wang);
25             byte[] reader=new byte[is.available()];
26             while(is.read(reader)!=-1)
27             {}
28             tv_filerawContent.setText(new String(reader));
29         }
30         catch(Exception e)
31         {
32             Toast.makeText(ReadRawFileActivity.this, "文件读取失败",
               Toast.LENGTH_LONG).show();
33         }
34         finally
35         {
36             if(is!=null)
37             {
38                 try
39                 {
40                     is.close();
41                 }
42                 catch(IOException e)
43                 {}
44             }
45         }
46     }
47 }
```

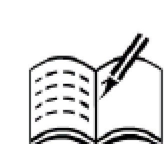
说明：

- 第 20 行：获取项目的资源。
- 第 21 行：定义一个 InputStream 输入流对象。
- 第 24 行：将输入流分配给一个原始资源的引用。
- 第 25 行：根据文件输入流的大小定义一个二进制数据数组，存放读取出的文件内容。
- 第 28 行：在 TextView 控件中显示文件内容。
- 第 32 行：当文件操作出现异常时，使用 Toast 显示错误提示信息。
- 第 40 行：当文件操作完毕后，关闭文件。

（10）创建 readxmlfile.xml 文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```






```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="读取 Res/xml/book.xml 文件:"
        android:textSize="20px"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/tv_xmlfilecontent"
        />
</LinearLayout>
```

说明：

在本布局文件中，先定义一个纵向的线性布局。在线性布局中依次定义两个 TextView 控件，第二个 TextView 用于显示资源文件的内容。

（11）首先在 res 下新建 xml 文件夹，用于存放自定义的 XML 文件，然后编写 XML 文件。编写代码如下：

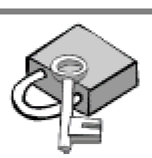
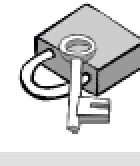
```
<BookList>
    <Book Category="Android 类">
        <bookitem name="Google Android 揭秘" author="W.Franl Ableson"/>
        <bookitem name="Android 应用开发揭秘" author="杨丰盛"/>
        <bookitem name="精通 Android 3" author="Satya Komatineni"/>
    </Book>
    <Book Category="程序设计类">
        <bookitem name="Java 编程思想" author="埃史尔"/>
        <bookitem name="Java 语言程序设计" author="Y.Daniel Liang"/>
        <bookitem name="JAVA 核心技术(卷 1)" author="Horstmann Gay S"/>
    </Book>
</BookList>
```

（12）创建 ReadXmlFileActivity 类文件 ReadXmlFileActivity.java。在这个类中，将读取 res\xml 下的 XML 文件，显示该文件的内容。编写代码如下：

```
package wyq.EX09_2;

import org.xmlpull.v1.XmlPullParser;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
public class ReadXmlFileActivity extends Activity {
    private TextView tv_xmlfilecontent;
    @Override
```

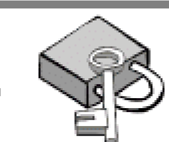



Note

```
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.readxmlfile);

    tv_xmlfilecontent=(TextView)findViewById(R.id.tv_xmlfilecontent);
    XmlPullParser xp=this.getResources().getXml(R.xml.book);
    String content="";
    try
    {
        while(xp.next()!=XmlPullParser.END_DOCUMENT)
        {
            String nodeName=xp.getName();
            String bookCategory=null;
            String bookName=null;
            String bookAuthor=null;

            if(nodeName!=null && nodeName.equals("Book"))
            {
                if(nodeName.equals("Book") && xp.getAttributeCount()!=-1 )
                {
                    bookCategory=xp.getAttributeValue(0);
                }
                if(bookCategory!=null)
                {
                    content=content+"图书类别:"+bookCategory+"\n";
                }
            }
            if(nodeName!=null && nodeName.equals("bookitem"))
            {
                for(int i=0;i<xp.getAttributeCount();i++)
                {
                    String attrName=xp.getAttributeName(i);
                    String attrValue=xp.getAttributeValue(i);
                    if(attrName.equals("name"))
                        bookName=attrValue;
                    if(attrName.equals("author"))
                        bookAuthor=attrValue;
                }
            }
            if(bookName!=null && bookAuthor!=null)
            {
                content=content+"书名:"+bookName+",作者:"+bookAuthor+"\n";
            }
        }
        this.tv_xmlfilecontent.setText(content);
    }
    catch(Exception e)
    {
```



```
61                Toast.makeText(ReadXmlFileActivity.this, "文件读取失败",
                  Toast.LENGTH_LONG).show();
62           }
63      }
64 }
```

说明：

- ❑ 第 18 行：定义一个 XmlPullParser 对象，来解析 XML 文件。要处理二进制 XML 资源，需要使用 XmlPullParser，这个类可以遍历 XML 树 SAX 样式。
- ❑ 第 22~58 行：遍历 XML 树。SAX 解析器为遇到的每个元素都提供了一个由整型数据表示的事件类型，如 START_TAG、END_TAG、START_DOCUMENT、END_DOCUMENT。使用 next()方法来检索当前的时间类型值并将它与类中的事件进行比较。END_DOCUMENT 表示文档结束事件。
  - ➢ 第 24 行：获取节点的名字。
  - ➢ 第 31~39 行：获取根节点的属性值。其中，第 31 行用于判断当前节点是否为 Book 节点（在本 XML 文件中为根节点），且判断该节点是否有属性。getAttributeCount()获取节点属性的个数，如果该节点没有属性，则返回-1。第 33 行获取节点的属性值。
  - ➢ 第 40~51 行：获取 XML 文件中其他节点的值。其中，第 44 行获取节点的属性名；第 45 行获取节点的属性值。

（13）创建 createxmlfile.xml 文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
      <LinearLayout
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:orientation="horizontal">
          <TextView
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="用户名:"
              />
          <EditText
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:id="@+id/edt_username"
              />
      </LinearLayout>
      <LinearLayout
       android:layout_width="fill_parent"
```

```
24         android:layout_height="wrap_content"
25         android:orientation="horizontal">
26             <TextView
27                 android:layout_width="wrap_content"
28                 android:layout_height="wrap_content"
29                 android:text="密　码:"
30                 />
31             <EditText
32                 android:layout_width="fill_parent"
33                 android:layout_height="wrap_content"
34                 android:id="@+id/edt_userpwd"
35                 />
36         </LinearLayout>
37         <Button
38             android:layout_width="fill_parent"
39             android:layout_height="wrap_content"
40             android:id="@+id/bt_save"
41             android:text="保存"
42             />
43 </LinearLayout>
```

说明：

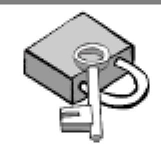
- 第 2~6 行：定义一个纵向的线性布局，其大小为整个屏幕。
- 第 7~21 行：在线性布局中嵌套一个横向的线性布局，包含一个 TextView 控件和一个 EditText 控件。
- 第 22~36 行：在线性布局中嵌套另一个横向的线性布局，包含一个 TextView 控件和一个 EditText 控件。
- 第 37~42 行：定义一个 Button 控件。

（14）创建 CreateXmlFileActivity 类文件 CreateXmlFileActivity.java。在这个类中，将实现在 XML 中保存用户名和密码，该 XML 文件将被保存在 data\data\[包名]\files 下。编写代码如下：

```
package wyq.EX09_2;

import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.StringWriter;
import org.xmlpull.v1.XmlSerializer;
import android.app.Activity;
import android.os.Bundle;
import android.util.Xml;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CreateXmlFileActivity extends Activity {
```

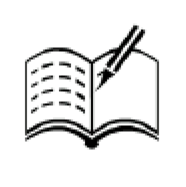





```
    private EditText edt_username;
    private EditText edt_userpwd;
    private Button bt_save;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.createxmlfile);
        edt_username=(EditText)findViewById(R.id.edt_username);
        edt_userpwd=(EditText)findViewById(R.id.edt_userpwd);
        bt_save=(Button)findViewById(R.id.bt_save);

        bt_save.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String username=edt_username.getText().toString();
                String userpwd=edt_userpwd.getText().toString();
                XmlSerializer serialer=Xml.newSerializer();
                StringWriter writer=new StringWriter();
                try
                {
                    serialer.setOutput(writer);
                    serialer.startDocument("utf-8",true);
                    serialer.startTag("", "UserList");
                    serialer.startTag("", "user");
                    serialer.attribute("", "username", username);
                    serialer.attribute("", "userpwd", userpwd);
                    serialer.endTag("", "user");
                    serialer.endTag("", "UserList");
                    serialer.endDocument();
                    OutputStream os=openFileOutput("user.xml",MODE_PRIVATE);
                    OutputStreamWriter oswriter=new OutputStreamWriter(os ,"GB2312");
                    oswriter.write(writer.toString());
                    oswriter.close();
                    os.close();
                }
                catch(Exception e)
                {
                    Toast.makeText(CreateXmlFileActivity.this, "用户信息保存不成功",
                    Toast.LENGTH_SHORT).show();
                }
            }

        });
    }
}
```


Note

说明：

第 28~53 行：为 Button 增加单击监听事件，实现 XML 文件的保存。

- 第 35 行：获取 XmlSerializer 对象。
- 第 39 行：设置输出流对象。
- 第 40 行：设置 XML 的文档开始。
- 第 41 行：设置 XML 的根节点。
- 第 42 行：设置 XML 的一个子节点。
- 第 43、44 行：设置 XML 子节点的属性与值。
- 第 45 行：设置 XML 子节点的结束。
- 第 46 行：设置 XML 根节点的结束。
- 第 47 行：设置 XML 文档的结束。
- 第 48 行：获取 XML 文件输出流的引用。
- 第 49 行：创建 OutputStreamWriter，以 GB2312 的编码格式往指定文件中写入信息，这样可以避免中文乱码的问题。OutputStreamWriter 是字符流通向字节流的桥梁。
- 第 50 行：写入 XML 数据。
- 第 51 行：关闭 OutputStreamWriter。
- 第 52 行：关闭输出流。

该实例运行结果如图 9-4~图 9-9 所示。用 File Explorer 查看文件的结果如图 9-10 所示。



图 9-4　EX09_2 运行界面



图 9-5　创建文件



图 9-6　读取文件

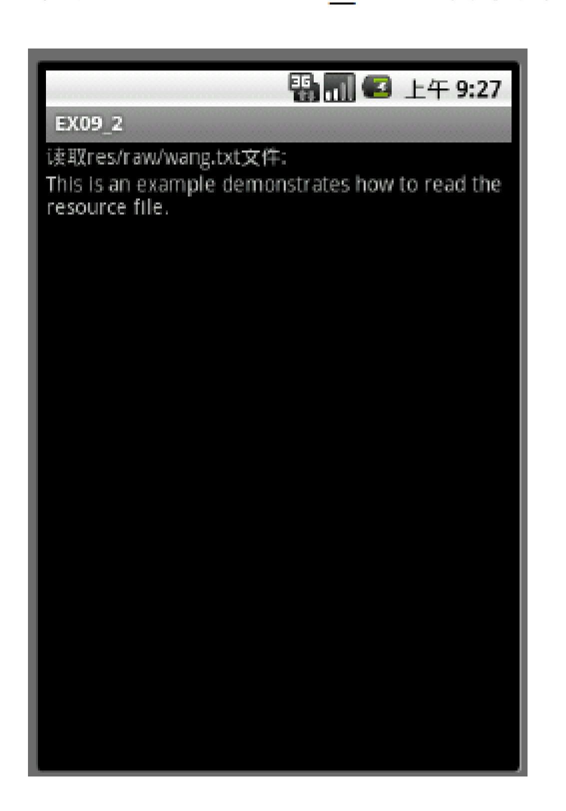


图 9-7　读取资源文件



图 9-8　读取 XML 文件

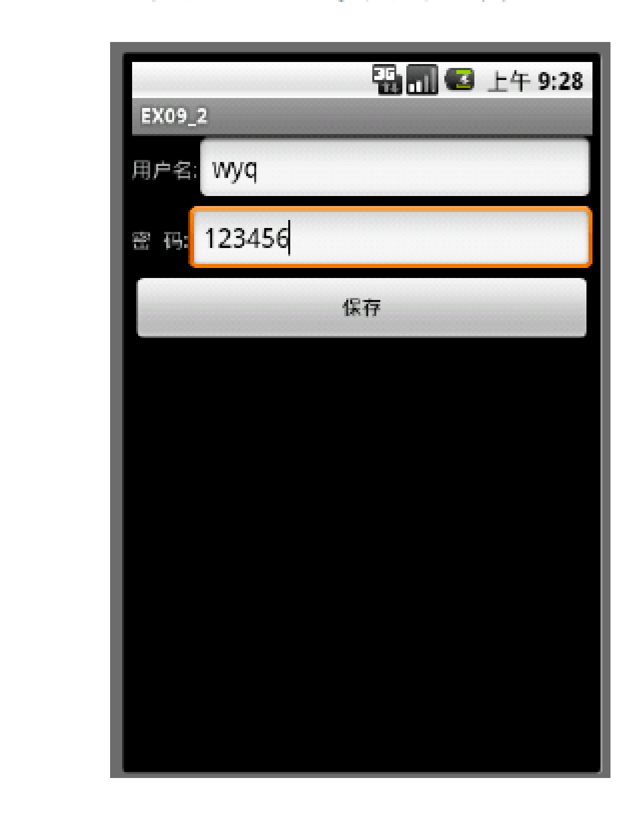
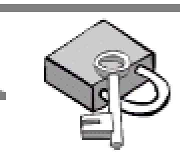


图 9-9　创建 XML 文件



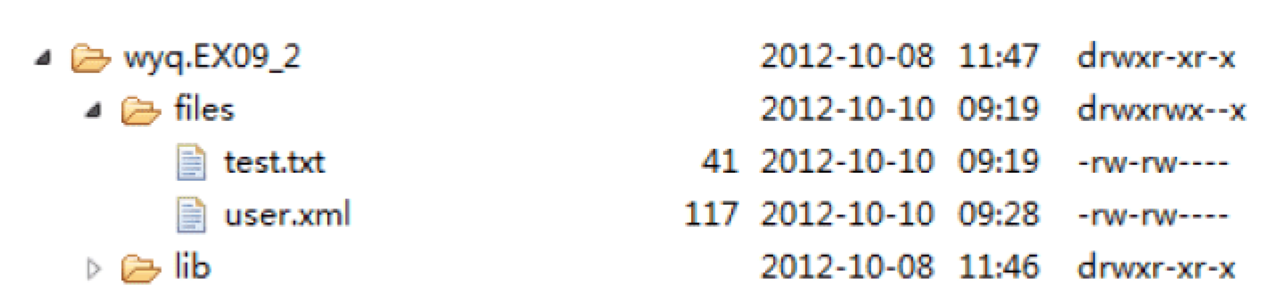
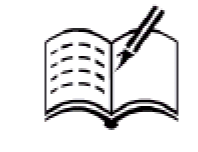


图 9-10 File Explorer 查看文件



9.3 数 据 库

数据库机制实际上也可以视为文件方式，Android 平台提供了创建和使用 SQLite 数据库的 API。与文件存取机制一样，每个数据库是其创建程序私有的，并不像普通桌面平台，数据库系统本身一般都是共享的，数据的访问权限是通过数据库管理系统来管理的。

SQLite 是一款轻型的数据库，是遵守 ACID 的关系式数据库管理系统，其设计目标是嵌入式的数据库，占用资源非常低。目前已经在很多嵌入式产品中使用了 SQLite，因为在嵌入式设备中，SQLite 可能只需要几百 KB 的内存。SQLite 能够支持 Windows、Linux、UNIX 等主流的操作系统，同时能够跟很多程序语言相结合，如 C#、PHP、Java 等，还有 ODBC 接口。与 MySQL、PostgreSQL 这两款世界著名的开源数据库管理系统相比，SQLite 的处理速度更快。SQLite 的第一个 Alpha 版本诞生于 2000 年 5 月，至今已有 13 个年头，现在 SQLite 3 已经发布。

在 Android 平台下，除了可以在 Android 程序中操作 SQLite 数据库之外，还可以在命令行模式下进行各种数据库的操作，包括表的各种操作，对数据的增加、删除、修改、查询等。本节除了通过实例介绍 SQLite 数据库的访问方式外，还将介绍如何在命令行模式下访问 SQLite 数据库。

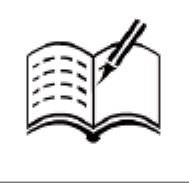
9.3.1 SQLite 数据库操作相关类简介

1．SQLiteOpenHelper 类

SQLiteOpenHelper 类是一个帮助类，用于创建数据库和管理数据库版本。使用该类，必须创建一个子类来实现其 onCreate(SQLiteDatabase)和 onUpgrade(SQLiteDatabase, int, int)方法。打开数据库进行操作必须保证数据库存在，如果不存在则创建它，并且对其进行必要的升级，维护其保持一个最佳的状态。使用本类提供的方法创建数据库是非常容易的，该类的常用方法如表 9-4 所示。

表 9-4 SQLiteOpenHelper 类常用方法

方 法 名 称	描 述
public SQLiteOpenHelper (Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	创建一个帮助对象，打开或者管理数据库。该方法通常快速返回。数据库并没有实际创建或打开，直到 getWritableDatabase()或 getReadableDatabase()其中一个被调用
Public synchronized void close ()	关闭任何打开的数据库对象



Note

续表

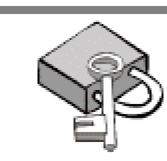
方 法 名 称	描 述
Public String getDatabaseName ()	返回正被打开的通过构造函数传递进来的 SQLite 数据库的名字
Public synchronized SQLiteDatabase getReadableDatabase ()	创建或打开一个数据库。这和 getWritableDatabase()返回的对象是同一个，除非一些因素要求数据库只能以只读的方式被打开，比如磁盘已满。在这种情况下，一个只读的数据库对象将被返回。如果这个问题被修改，将来调用 getWritableDatabase()就可能成功，而这时只读的数据库对象将被关闭，并且读写对象将被返回
Public synchronized SQLiteDatabase getWritableDatabase ()	创建或打开一个数据库，用于读写。该方法第一次被调用时，数据库被打开，并且 onCreate(SQLiteDatabase)、onUpgrade(SQLiteDatabase,int,int)或 onOpen(SQLiteDatabase)将被调用
Public abstract void onCreate (SQLiteDatabase db)	当第一次创建数据库时调用。表格的创建和初始化表格的个数在这里完成

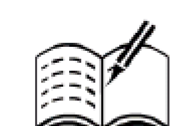
2．SQLiteDatabase 类

SQLiteDatabase 用于管理 SQLite 数据库，对数据库中的数据进行增加、修改、删除、查询、执行 SQL 命令，并执行其他常见的数据库管理任务。该类的常用方法如表 9-5 所示。

表 9-5　SQLiteDatabase 类常用方法

方 法 名 称	参 数 说 明	方 法 描 述
（1）public void beginTransaction() （2）public void beginTransactionWithListener (SQLiteTransactionListener transactionListener)	transactionListener：通知在事务开始时提交或回滚调用的监听器	开始事务
public void endTransaction()		结束事务
（1）public void execSQL(String sql, Object[] bindArgs) （2）public void execSQL(String sql)	sql：SQL 语句 bindArgs：SQL 语句的参数	执行一个非查询的 SQL 语句
public long insert(String table, String nullColumnHack, ContentValues values)	table：表名 values：插入的数据，类似于 Map。通过键-值对的形式存储值，键是表的列名，而值是该列的字段值	插入数据
public int delete(String table, String whereClause, String[] whereArgs)	table：表名 whereClause：删除数据的条件，如果为 null，则删除表中所有的数据	删除数据
public int update(String table, ContentValues values, String whereClause, String[] whereArgs)	table：表名 values：修改的数据 whereClause：修改数据的条件，如果为 null，则修改表中所有的数据	修改数据





续表

方法名称	参数说明	方法描述
（1）public Cursor query(Boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit) （2）public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy) （3）public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)	distinct：如果为 true，则在查询结果中去掉重复的行 table：表名 columns：查询列的列表 selection：查询条件 selectArgs：查询条件参数 groupBy：分组字段，格式化为一个 SQL 的 GROUP BY 子句 having：格式化为一个 SQL 的 HAVING 子句 orderBy：格式化为一个 SQL 的 ORDER BY 子句 limit：返回的行数	查询数据

9.3.2 SQLite 数据库使用实例

在 9.3.1 节中，介绍了 SQLite 数据库操作的相关类，本节将通过一个实例介绍 SQLite 数据库的使用方法。在本实例中，将创建一个数据库 Db_People，在该数据库中创建一张表 tb_people，表结构如表 9-6 所示。运行本实例程序时，使用 ListView 控件显示 tb_people 表中的数据，并在 ListView 控件上绑定上下文菜单，在上下文菜单中，可以对 ListView 选中项进行修改和删除，并可以通过选项菜单向 tb_people 表中插入数据。

表 9-6 tb_people 表结构

列名	数据类型	描述	说明
_id	integer	编号	primary key autoincrement
name	varchar(20)	姓名	
phone	varchar(12)	电话	
mobile	varchar(12)	手机	
email	varchar(30)	电子信箱	

本实例的开发步骤如下：

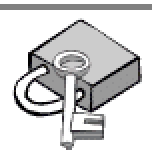
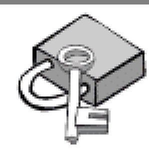
（1）创建项目 EX09_3。

（2）编写数据库帮助类 DbHelper 文件 DbHelper.java，编写代码如下：

```
package wyq.EX09_3;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class DbHelper extends SQLiteOpenHelper {
```

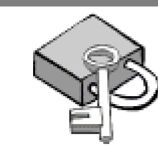

```
9
10     public DbHelper(Context context,String name,CursorFactory factory,int version)
11     {
12         super(context,name,factory,version);
13     }
14     @Override
15     public void onCreate(SQLiteDatabase db) {
16         // TODO Auto-generated method stub
17         db.execSQL("create table if not exists tb_people" +
18                 " (_id integer primary key autoincrement, " +
19                 "name varchar(20), " +
20                 "phone varchar(12), " +
21                 "mobile varchar(12), " +
22                 "email varchar(30)) ");
23     }
24     @Override
25     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
26         // TODO Auto-generated method stub
27     }
28 }
```

说明：

- 第 10~13 行：重写 DbHelper 的构造函数，回调父函数的 super(context,name,factory, version)。
- 第 15~23 行：重写 OnCreate()函数。在本函数中，创建数据库中的表。因为在本例中要使用 SimpleCursorAdapter，而 SimpleCursorAdapter 要求表的主键为_ID，否则会出现“不存在_ID 列”的错误，所以本表的主键列为_ID。

（3）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="所有联系人： "
        android:textSize="15px"
        />
     <LinearLayout
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content">
           <TextView
          android:layout_width="40px"
          android:layout_height="wrap_content"
```




```
            android:text="编号"
        />
        <TextView
            android:layout_width="50px"
            android:layout_height="wrap_content"
            android:text="姓名"
        />
            <TextView
            android:layout_width="80px"
            android:layout_height="wrap_content"
            android:text="电话"
        />
        <TextView
            android:layout_width="80px"
            android:layout_height="wrap_content"
            android:text="手机"
        />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="电子信箱"
        />
    </LinearLayout>
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:id="@+id/list_people"
        />
</LinearLayout>
```

说明：

在本布局文件中，首先定义一个屏幕大小的纵向线性布局，包含一个 TextView 控件、一个嵌套的横向线性布局和一个 ListView 控件。横向的线性布局中包含 5 个 TextView 控件，用于显示用户信息列表的表头。ListView 控件用于显示从数据库中读取出的数据。

（4）编写 ListView 的 Item 显示布局文件 peoplelist.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="horizontal">
  <TextView
        android:id="@+id/id"
        android:layout_width="40px"
        android:layout_height="wrap_content"
        />
     <TextView
```

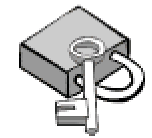
```
        android:id="@+id/name"
        android:layout_width="50px"
        android:layout_height="wrap_content"
        />
    <TextView
        android:id="@+id/phone"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        />
    <TextView
        android:id="@+id/mobile"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        />
    <TextView
        android:id="@+id/email"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

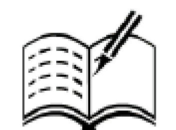
（5）修改主 Activity 的类文件 FirstActivity.java。在这个 Activity 中，首先使用 ListView 显示数据库中所有的数据，在 ListView 中绑定了上下文菜单，长按某一项，可以对该项进行修改和删除；通过选项菜单可以增加数据和退出程序。编写代码如下：

```
package wyq.EX09_3;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.AdapterView.AdapterContextMenuInfo;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private ListView list_people;
    private DbHelper dbhelper;
    private SQLiteDatabase db;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```





Note

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        list_people=(ListView)findViewById(R.id.list_people);

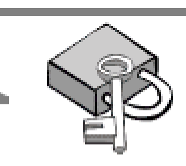
        dbhelper=new DbHelper(this, "Db_People",null, 1);
        db=dbhelper.getReadableDatabase();
        Cursor c=db.query("tb_people", new String[]
                    {"_id","name","phone","mobile","email"}, null, null,null,null,null);
        SimpleCursorAdapter adapter=new SimpleCursorAdapter(this,
                R.layout.peoplelist,
                c,
                new String[]{"_id","name","phone","mobile","email"},
                new int[]{R.id.id,R.id.name,R.id.phone,R.id.mobile,R.id.email});
        this.list_people.setAdapter(adapter);

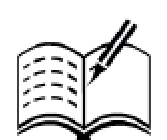
        this.registerForContextMenu(list_people);
    }
    @Override
    public    oolean onCreateOptionsMenu(Menu menu) {
        // TODO Auto-generated method stub
        menu.add(Menu.NONE, Menu.FIRST + 1, 1, "添加")
            .setIcon(android.R.drawable.ic_menu_add);
        menu.add(Menu.NONE, Menu.FIRST + 2, 2, "退出")
            .setIcon(android.R.drawable.ic_menu_delete);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
       switch (item.getItemId())
       {
               case Menu.FIRST + 1:Intent intent=new Intent();
                     intent.setClass(FirstActivity.this, AddPeopleActivity.class);
                                startActivity(intent);
                                break;
           case Menu.FIRST + 2:finish();
                                break;
       }
       return super.onOptionsItemSelected(item);
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
    {
        // TODO Auto-generated method stub
        menu.setHeaderIcon(R.drawable.icon);
            menu.add(0,3,0, "修改");
            menu.add(0,4,0, "删除");
    }
```

```
71     @Override
72     public boolean onContextItemSelected(MenuItem item)
73     {
74         AdapterContextMenuInfo menuInfo = (AdapterContextMenuInfo) item.getMenuInfo();
75         // TODO Auto-generated method stub
76         switch(item.getItemId())
77         {
78             case 3:
79                 String name = ((TextView) menuInfo.targetView.findViewById
                        (R.id.name)).getText().toString();
80                 String phone = ((TextView) menuInfo.targetView.findViewById
                        (R.id.phone)).getText().toString();
81                 String mobile = ((TextView) menuInfo.targetView.findViewById
                        (R.id.mobile)).getText().toString();
82                 String email = ((TextView) menuInfo.targetView.findViewById
                        (R.id.email)).getText().toString();
83                 Intent intent=new Intent();
84                 intent.setClass(FirstActivity.this, AddPeopleActivity.class);
85                 Bundle bundle=new Bundle();
86                 bundle.putLong("id", menuInfo.id);
87                 bundle.putString("name",name);
88                 bundle.putString("phone",phone);
89                 bundle.putString("mobile", mobile);
90                 bundle.putString("email", email);
91                 intent.putExtras(bundle);
92                 startActivity(intent);
93                 break;
94             case 4:
95                 dbhelper=new DbHelper(this, "Db_People",null, 1);
96                 db=dbhelper.getWritableDatabase();
97                 db.delete("tb_people","_id=?", new String[]{menuInfo.id+""});
98                 break;
99         }
100        return true;
101    }
102 }
```

说明：

- 第 30 行：创建 Db_People 数据库。
- 第 31 行：使用 getReadableDatabase()打开数据库。
- 第 32 行：定义一个游标，从 tb_people 表中查询数据。
- 第 33 行：定义 SimpleCursorAdapter 对象，使用的资源文件为 R.layout.peoplelist，用第 32 行定义的游标作为适配器的数据源。
- 第 38 行：将第 33 行定义的适配器设置为 ListView 控件的适配器。
- 第 40 行：为 ListView 控件注册上下文菜单。
- 第 43~48 行：创建 Menu 选项菜单。
- 第 50~62 行：为选项菜单增加处理事件。

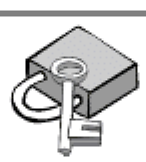
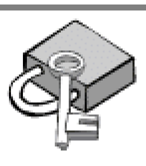




- 第 64~70 行：创建上下文菜单。
- 第 72~101 行：为上下文菜单增加处理事件。
  - 第 74 行：获得 AdapterContextMenuInfo，以此来获得选择的 ListView 项目。
  - 第 79~82 行：获取所选择的 ListView 中 Item 中各项的值。因为 ListView 中各项的值是在 TextView 控件中显示的，获取到相应控件，就可以得到相应的值。
  - 第 83~84 行：生成一个 Intent 对象。
  - 第 85~91 行：为 Intent 绑定要传输的值。第 86 行中，menuInfo.id 获取所选择的 ListView 项目的 ID。
  - 第 92 行：打开 Intent 对象中设置的另一个 Activity。
  - 第 96 行：使用 getWritableDatabase()打开数据库。
  - 第 97 行：从数据库中将选择的 ListView 项目删除。menuInfo.id 为所选择的 ListView 项目的 ID。

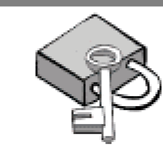
（6）编写布局文件 addpeople.xml，作为增加、修改数据 Activity 的布局文件，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
      <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal">
          <TextView
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:text="用户名"
              android:layout_weight="2"
              />
          <EditText
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:id="@+id/edt_name"
              android:layout_weight="1"
              />
      </LinearLayout>
      <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal">
          <TextView
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
```

Note

```
            android:text="联系电话"
            android:layout_weight="2"
            />
        <EditText
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/edt_phone"
            android:layout_weight="1"
            />
    </LinearLayout>
    <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="手机"
            android:layout_weight="2"
            />
        <EditText
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/edt_mobile"
            android:layout_weight="1"
            />
    </LinearLayout>
    <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="电子信箱"
            android:layout_weight="2"
            />
        <EditText
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/edt_email"
            android:layout_weight="1"
            />
    </LinearLayout>
    <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
        <Button
```




```
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/bt_save"
                android:text="保存"
                android:layout_weight="1"
                />
            <Button
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/bt_cancel"
                android:text="取消"
                android:layout_weight="1"
                />
        </LinearLayout>
</LinearLayout>
```

说明：

在本布局文件中，定义一个大小为整个屏幕的纵向布局，包含嵌套在其中的 4 个横向线性布局以及一个 Button 控件。每一个横向线性布局包含一个 TextView 和一个 EditText 控件，用于输入数据；Button 控件用于产生命令，对数据进行添加或者修改。

（7）创建 AddPeopleActivity 类文件 AddPeopleActivity.java。在这个 Activity 中，根据从上一个 Activity 中是否有传递数据，来判断是修改所传递数据，还是可以向表中插入数据。编写代码如下：

```
package wyq.EX09_3;

import android.app.Activity;
import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddPeopleActivity extends Activity {
    private EditText edt_name;
    private EditText edt_phone;
    private EditText edt_mobile;
    private EditText edt_email;
    private Button bt_save;
    String name,phone,mobile,email;
    DbHelper dbhelper;
    SQLiteDatabase db;
    Bundle bundle;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
```

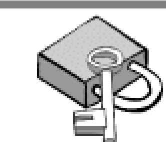

Note

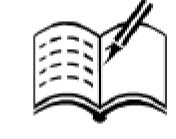
```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.addpeople);

        edt_name=(EditText)findViewById(R.id.edt_name);
        edt_phone=(EditText)findViewById(R.id.edt_phone);
        edt_mobile=(EditText)findViewById(R.id.edt_mobile);
        edt_email=(EditText)findViewById(R.id.edt_email);
        bt_save=(Button)findViewById(R.id.bt_save);

        bundle=this.getIntent().getExtras();
        if(bundle!=null)
        {
                edt_name.setText(bundle.getString("name"));
                edt_phone.setText(bundle.getString("phone"));
                edt_mobile.setText(bundle.getString("mobile"));
                edt_email.setText(bundle.getString("email"));
        }
        bt_save.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                name=edt_name.getText().toString();
                phone=edt_phone.getText().toString();
                mobile=edt_mobile.getText().toString();
                email=edt_email.getText().toString();

                ContentValues value=new ContentValues();
                value.put("name", name);
                value.put("phone", phone);
                value.put("mobile", mobile);
                value.put("email", email);
                DbHelper dbhelper = new DbHelper
                      (AddPeopleActivity.this, "Db_People",null, 1);
                SQLiteDatabase db=dbhelper.getWritableDatabase();
                long status;
                if(bundle!=null)
                {
                    status=db.update("tb_people", value, "_id=?",
                      new String[]{bundle.getLong("id")+""});
                }
                else
                {
                    status=db.insert("tb_people", null, value);
                }
                if(status!=-1)
                {
                Toast.makeText(AddPeopleActivity.this, "保存成功",
                      Toast.LENGTH_LONG).show();
```






```
71                  }
72                  else
73                  {
74                      Toast.makeText(AddPeopleActivity.this, "保存失败",
                            Toast.LENGTH_LONG).show();
75                  }
76              }
77          });
78      }
79 }
```

说明：

- 第 34 行：获取 Intent 绑定的数据。
- 第 35~41 行：如果要修改数据，则将 Intent 中绑定的数据显示在各个 EditText 控件中进行编辑。
- 第 42~47 行：为 Button 控件增加单击监听事件，用于向数据库保存数据。在本事件中，根据从上一个 Activity 是否有传递值，来判断对数据库的操作是更新还是插入。
- 第 52~56 行：生成 ContentValues，用于存放向数据库保存的数据。
- 第 57、58 行：获取数据库的引用后，打开数据库。
- 第 60~67 行：如果 bundle 为 null，说明没有从上一个 Activity 传输数据，则将数据插入到数据库；如果不为 null，则修改数据库中的数据。第 62 行更新数据库的数据；第 66 行向数据库插入数据。
- 第 68~76 行：根据插入数据和更新数据的返回值，判断数据是否保存成功，并进行提示。

本实例运行结果如图 9-11~图 9-13 所示。



图 9-11　EX09_3 运行结果



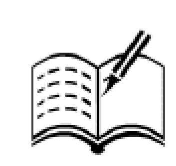
图 9-12　增加信息



图 9-13　删除和修改信息

9.3.3　使用命令行操作 SQLite

在运行 EX09_3 后，可以通过 DDMS，使用 File Explorer 查看 Db_People 数据库文件，



Note

该数据库文件存放在 data\data\wyq.EX09_3\databases 下。Android 平台基于 Linux 核心，除了可以使用前台程序操作 SQLite 数据库之外，还可以通过命令行操作数据库。本节将介绍如何使用命令行访问数据库，步骤如下：

（1）运行 Android 模拟器。

（2）在 Windows 的命令行模式下，输入命令 adb shell，登录到设备的 shell，进入到 Linux 的命令行。当 adb shell 之后的提示字符为“#”时，表示使用者为 root（最大权限）。如果出现“adb 不是内部或外部命令”的提示，说明 Android 的环境变量没有配置正确。

（3）输入命令 sqlite3 data/data/wyq.EX09_3/databases/Db_People。其中，sqlite3 用于打开数据库，data\data\wyq.EX09_3\databases\Db_People 则是数据库文件的路径，如图 9-14 所示。

```
管理员: C:\Windows\system32\cmd.exe - adb shell
C:\Users\Administrator>adb shell
# ls
ls
sqlite_stmt_journals
config
cache
sdcard
d
etc
system
sys
sbin
proc
init.rc
init.goldfish.rc
init
default.prop
data
root
dev
# sqlite3 data/data/wyq.EX09_3/databases/Db_People
sqlite3 data/data/wyq.EX09_3/databases/Db_People
SQLite version 3.5.9
Enter ".help" for instructions
sqlite>
```

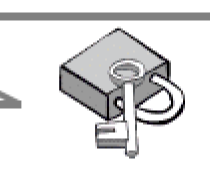
图 9-14　命令行操作 SQLite 数据库

（4）打开数据库后，可以在命令行输入各种命令来操作数据库。例如，输入.schema 可以查看数据库中所有表的结构；输入 create table 可以在数据库中创建表等。

（5）数据库查看完毕后，输入.exit 退出 SQLite 3 程序，关闭数据库的访问。

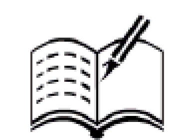
9.4　ContentProvider 类

在 Android 中，每个应用程序都在各自的进程中运行，并且存储于其中的数据和文件默认不能有其他应用程序访问。当然可以通过设置正确的权限，将首选项和文件设置为供不同的应用程序使用，但是对于相互了解对方详细信息的相关应用程序来说有一定的局限性。通过 ContentProvider 类，可以发布和公开一个特定的数据类型，提供增加、修改、删除和查询的操作，其他应用程序可以利用该应用程序提供的 ContentProvider 类执行数据的增加、修改、删除和查询操作，并且不需要对方提供路径、资源，甚至谁提供了什么内容



都不需要知道。

Android 中标准的 ContentProvider 实例就是联系人列表，应用程序开发人员可以在任何应用程序中使用特定的 URI（Content://contacts/people）来访问联系人进行各种操作。在 Activity 和 ContentResolver 类提供了很多用于存储和查询数据的方法。本节将通过实例来介绍 ContentProvider 的使用。



9.4.1 ContentProvider 类简介

1. URI

每个 ContentProvider 都需要公开一个唯一的 CONTENT_URI，能够表示当前所处理的内容类型。可以通过两种方式使用这个 URI 来查询数据，即单独使用和结合使用，如表 9-7 所示。

表 9-7 URI

URI	描　述
content://authority/data	从已注册为处理 content://authority 的处理程序处返回所有数据的列表
content://authority/data/ID	从已注册为处理 content://authority 的处理程序处返回指定 ID 的数据列表

以本节将要使用的 URI content://wyq.EX09_3.Db_People/tb_people 为例，URI 由以下几部分组成。

（1）标准前缀：用来说明由一个 ContentProvider 控制这些数据，此部分无法改变。

（2）URI 的标识：定义了是哪个 ContentProvider 提供这些数据。该标识在<provider>元素的 authorities 属性中说明：

```
<provider name=".PeopleProvider" authorities="wyq.EX09_3.Db_People"/ >
```

（3）路径：ContentProvider 使用这些路径来确定当前需要什么类型的数据，URI 中可能不包括路径，也可能包括多个。

（4）如果 URI 中包含 ID，表示需要获取的记录的 ID；如果没有 ID，就表示返回全部。

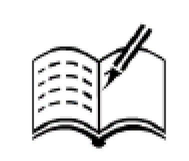
由于 URI 通常比较长，而且容易出错且难以理解。所以，在 Android 中定义了一些辅助类和常量来代替这些长字符串，如 People.CONTENT_URI。

2. ContentProvider 类

Android 提供了 ContentProvider，一个程序可以通过实现一个 ContentProvider 的抽象接口将自己的数据完全暴露出去，而且 ContentProvider 是以类似数据库中表的方式将数据暴露，也就是说，ContentProvider 就像一个“数据库”。那么外界获取其提供的数据，也就与从数据库中获取数据的操作基本一样，只不过是采用 URI 来表示外界需要访问的“数据库”。至于如何从 URI 中识别出外界需要的是哪个“数据库”，这就是 Android 底层需要做的事情了。ContentProvider 向外界提供数据操作的接口如表 9-8 所示。

表 9-8　ContentProvider 接口

函　　数	说　　明
query(Uri, String[], String, String[], String)	查询数据
insert(Uri, ContentValues)	插入数据
update(Uri, ContentValues, String, String[])	修改数据
delete(Uri, String, String[])	删除数据



Note

实现 ContentProvider 的过程如下：

（1）生成一个继承于 ContentProvider 的子类，实现相应的方法。

（2）ContentProvider 通常需要对外提供 CONTENT_URI、URI_AUTHORITY 和对外的数据字段常量等。

（3）提供 UriMatcher，用来判断外部传入的 URI 是否带有 ID。

（4）根据自己保存数据的具体实现，来重写 ContentProvider 的 query()、delete()、update()、insert()、onCreate()、getType()方法。

（5）在 AndroidMainfest.xml 中声明该 ContentProvider 类。

3．ContentResolver 类

外界的程序通过 ContentResolver 接口可以访问 ContentProvider 提供的数据。在 Activity 中通过 getContentResolver()可以得到当前应用的 ContentResolver 实例。

ContentResolver 提供的接口和 ContentProvider 中需要实现的接口对应，主要有以下几个，如表 9-9 所示。

表 9-9　ContentResolver 接口

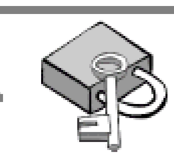
接 口 函 数	说　　明
final Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,String sortOrder)	通过 Uri 进行查询，返回一个 Cursor
final Uri insert(Uri url, ContentValues values)	将一组数据插入到 Uri 指定的地方
final int update（Uri uri, ContentValues values, String where, String[] selectionArgs）	更新 Uri 指定位置的数据
final int delete（Uri url, String where, String[] selectionArgs）	删除指定 Uri 并且符合一定条件的数据

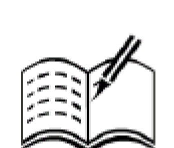
9.4.2　ContentProvider 使用实例

本节将通过实例来介绍如何实现一个 ContentProvider，及如何在另外一个项目中使用该 ContentProvider。

在本实例中，首先在项目 EX09_3 中实现该项目的 ContentProvider，然后在项目 EX09_4 中对该数据库的数据进行访问操作。项目 EX09_4 的界面、功能与 EX09_3 完全相同，所以在本节的代码中主要突出体现代码的不同之处。

本实例的开发步骤如下：



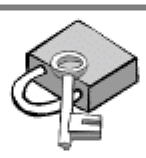
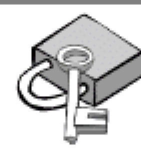


（1）在项目 EX09_3 中新建 PeopleProvider.java 类文件，实现该项目的 ContentProvider，编写代码如下：

```
package wyq.EX09_3;

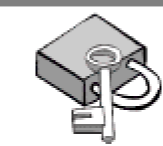
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.text.TextUtils;

public class PeopleProvider extends ContentProvider {
    private static final int ITEMS=1;
    private static final int ITEM_ID=2;
    public static final String DbName="Db_People";
    public static final String TableName="tb_people";
    DbHelper dbhelper;
    SQLiteDatabase db;
    public static final String CONTENT_ITEMS_TYPE =
        "vnd.android.cursor.items/wyq.EX09_3.Db_People";
    public static final String CONTENT_ITEMID_TYPE =
        "vnd.android.cursor.itemid/wyq.EX09_3.Db_People";
    public static final Uri CONTENT_URI =
        Uri.parse("content://wyq.EX09_3.Db_People/tb_people");
    private static final UriMatcher sMatcher;
    static
    {
        sMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        sMatcher.addURI("wyq.EX09_3.Db_People", TableName, ITEMS);
        sMatcher.addURI("wyq.EX09_3.Db_People", TableName+"/#",ITEM_ID);
    }
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        db = dbhelper.getWritableDatabase();
        int count = 0;
        switch (sMatcher.match(uri)) {
        case ITEMS:
                count = db.delete("tb_people",selection, selectionArgs);
                break;
        case ITEM_ID:
                String id = uri.getPathSegments().get(1);
                count = db.delete("tb_people", "_ID="+id+
                 (!TextUtils.isEmpty("_ID=?")?"AND("+selection+')': ""), selectionArgs);
                break;
        default:
                throw new IllegalArgumentException("Unknown URI"+uri);
```

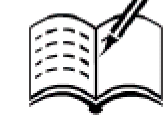




Note

```
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }
    @Override
    public String getType(Uri uri) {
        // TODO Auto-generated method stub
        switch (sMatcher.match(uri)) {
        case ITEMS:
            return CONTENT_ITEMS_TYPE;
        case ITEM_ID:
            return CONTENT_ITEMID_TYPE;
        default:
            throw new IllegalArgumentException("Unknown URI"+uri);
        }
    }
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        db = dbhelper.getWritableDatabase();
        long rowId;
        if(sMatcher.match(uri)!=ITEMS){
            throw new IllegalArgumentException("Unknown URI"+uri);
        }
        rowId = db.insert("tb_people","_ID",values);
        if(rowId>0)
        {
            Uri noteUri=ContentUris.withAppendedId(CONTENT_URI, rowId);
            getContext().getContentResolver().notifyChange(noteUri, null);
            return noteUri;
        }
        throw new IllegalArgumentException("Unknown URI"+uri);
    }
    @Override
    public boolean onCreate() {
        // TODO Auto-generated method stub
        dbhelper=new DbHelper(this.getContext(),"Db_People",null, 1);
        return true;
    }
    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {
        db = dbhelper.getReadableDatabase();
        Cursor c;
        switch (sMatcher.match(uri)) {
        case ITEMS:
                c = db.query("tb_people", projection, selection, selectionArgs, null, null, null);
                break;
        case ITEM_ID:
                String id = uri.getPathSegments().get(1);
```



```
92                  c = db.query("tb_people", projection, "_ID="+id+(!TextUtils.isEmpty
                  (selection)? "AND("+selection+')': ""),selectionArgs, null, null, sortOrder);
93              break;
94          default:
95                  throw new IllegalArgumentException("Unknown URI"+uri);
96          }
97          c.setNotificationUri(getContext().getContentResolver(), uri);
98          return c;
99      }
100     @Override
101     public int update(Uri uri, ContentValues values, String selection,
102             String[] selectionArgs) {
103         db = dbhelper.getWritableDatabase();
104         int count = 0;
105         switch (sMatcher.match(uri)) {
106         case ITEMS:
107                 count = db.update("tb_people",values,selection, selectionArgs);
108                 break;
109         case ITEM_ID:
110                 String id = uri.getPathSegments().get(1);
111                 count = db.update("tb_people", values, "_ID="+id+
                (!TextUtils.isEmpty("_ID=?")?"AND("+selection+')': ""), selectionArgs);
112                 break;
113         default:
114                 throw new IllegalArgumentException("Unknown URI"+uri);
115         }
116         getContext().getContentResolver().notifyChange(uri, null);
117         return count;
118     }
119 }
```



Note

说明：

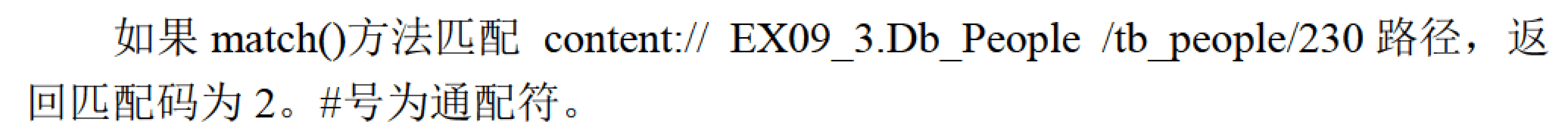
- 第 13、14 行：定义两个整型常量，用于表示 UriMatcher 匹配的结果。
- 第 15、16 行：定义两个与数据库相关的常量来定义要使用的数据库名和表名。
- 第 17 行：定义 DbHelper 对象。
- 第 18 行：定义一个 SQLiteDatabase 对象，用于存储和检索提供程序处理的数据。
- 第 19、20 行：定义两个特定的 MIME 条目，并将它与单条目路径及多条目路径结合起来，表示单条数据的 MIME 类型和复数条数据的 MIME 类型。
- 第 21 行：定义 URI，用于发布。URI 的结构见 9.4.1 节。
- 第 22~28 行：定义 UriMatcher，用于匹配 URI。其用法如下。

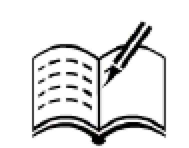
  首先，把需要匹配 URI 路径全部注册，如下所示：

```
UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
```

常量 UriMatcher.NO_MATCH 表示不匹配任何路径的返回码为-1。

```
addUri("wyq.EX09_3.Db_People", TableName, ITEMS);
```





添加需要匹配 URI，如果匹配就会返回匹配码。如果 match()方法匹配 content:// EX09_3.Db_People /tb_people 路径，返回匹配码为 1。

```
addUri("wyq.EX09_3.Db_People", TableName+"/#",ITEM_ID);
```

如果 match()方法匹配 content:// EX09_3.Db_People /tb_people/230 路径，返回匹配码为 2。#号为通配符。

注册完需要匹配的 URI 后，就可以使用 uriMatcher.match(uri)方法对输入的 URI 进行匹配，如果匹配就返回匹配码，匹配码是调用 addUri()方法传入的第 3 个参数，假设匹配 content:// EX09_3.Db_People /tb_people 路径，返回的匹配码为 1。

- 第 30~46 行：实现 delete()方法，提供删除数据的方法。处理过程为，将传入的 URI 与单一元素或这个元素集进行匹配，然后对数据库对象调用各自的删除方法。在这些方法结束部分，通知侦听程序数据已更改。
- 第 48~58 行：实现 getType()方法。提供程序将使用该方法来解析各个传入的 URI，以确定它是否支持以及当前调用所请求的数据类型。此处返回的字符串是在类中定义的常量。
- 第 60~74 行：实现 insert()方法，提供插入数据的方法。处理过程为，调用数据库插入方法并返回生成 URI 和新记录的附加 ID。完成插入操作之后，针对 ContentResolver 的通知系统将开始运行。在这里，由于对数据进行了修改，因此将所生成的事件通知给 ContentResolver，以便更新任何已注册的监听程序。
- 第 76~80 行：实现 OnCreate()方法，定义 DbHelper 对象。
- 第 82~99 行：实现 query()方法，提供查询数据的方法。处理过程为，将传入的 URI 与单一元素或这个元素集进行匹配，然后对数据库对象调用各自的查询方法，并获取要返回的 Cursor 句柄。在查询方法的结束部分，使用 setNotificationUri()方法监视 URI 的更改，可以跟踪 Cursor 数据条目何时发生了变更。
- 第 101~118 行：实现 update()方法，提供数据更新的方法。处理过程与 delete()方法类似。

（2）修改项目 EX09_3 的 AndroidManifest 文件，在 Application 节点之间增加以下代码：

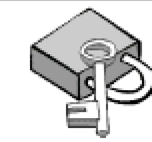
```
<provider name=".PeopleProvider" authorities="wyq.EX09_3.Db_People"/ >
```

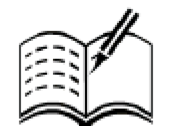
（3）新建项目 EX09_4，修改该项目主 Activity 的类文件 FirstActivity.java。在这个 Activity 中，首先使用 ListView 显示数据库中所有的数据，在 ListView 中绑定了上下文菜单，长按某一项，可以对该项进行修改和删除；通过选项菜单可以增加数据和退出程序。编写代码如下：

```
package wyq.EX09_4;

import android.app.Activity;
import android.content.ContentResolver;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
```

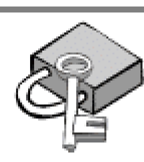
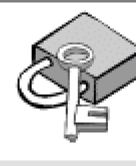






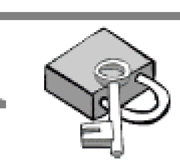
```
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.AdapterView.AdapterContextMenuInfo;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private ListView list_people;
    private ContentResolver contentResolver;
    private Uri CONTENT_URI =
            Uri.parse("content://wyq.EX09_3.Db_People/tb_people");
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        contentResolver = this.getContentResolver();
        list_people=(ListView)findViewById(R.id.list_people);
        Cursor c=contentResolver.query(CONTENT_URI,
                new String[]{"_id","name","phone","mobile","email"}, null, null,null);
        SimpleCursorAdapter adapter=new SimpleCursorAdapter(this,
                R.layout.peoplelist,
                c,
                new String[]{"_id","name","phone","mobile","email"},
                new int[]{R.id.id,R.id.name,R.id.phone,R.id.mobile,R.id.email});
        this.list_people.setAdapter(adapter);
        this.registerForContextMenu(list_people);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // TODO Auto-generated method stub
        menu.add(Menu.NONE, Menu.FIRST + 1, 1, "添加")
                    .setIcon(android.R.drawable.ic_menu_add);
        menu.add(Menu.NONE, Menu.FIRST + 1, 2, "退出")
                    .setIcon(android.R.drawable.ic_menu_delete);
        return true;    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
       switch (item.getItemId())
       {
            case Menu.FIRST + 1:Intent intent=new Intent();
                    intent.setClass(FirstActivity.this, AddPeopleActivity.class);
                    startActivity(intent);
```


Note

```
                break;
            case Menu.FIRST + 2:finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
    {
        // TODO Auto-generated method stub
        menu.setHeaderIcon(R.drawable.icon);
        menu.add(0,3,0, "修改");
        menu.add(0,4,0, "删除");
    }
    @Override
    public boolean onContextItemSelected(MenuItem item)
    {
        AdapterContextMenuInfo menuInfo = (AdapterContextMenuInfo) item.getMenuInfo();
        // TODO Auto-generated method stub
        switch(item.getItemId())
        {
            case 3:
                String name = ((TextView) menuInfo.targetView
                                .findViewById(R.id.name)).getText().toString();
                String phone = ((TextView) menuInfo.targetView
                                .findViewById(R.id.phone)).getText().toString();
                String mobile = ((TextView) menuInfo.targetView
                                .findViewById(R.id.mobile)).getText().toString();
                String email = ((TextView) menuInfo.targetView
                                .findViewById(R.id.email)).getText().toString();
                Intent intent=new Intent();
                intent.setClass(FirstActivity.this, AddPeopleActivity.class);
                Bundle bundle=new Bundle();
                bundle.putLong("id", menuInfo.id);
                bundle.putString("name",name);
                bundle.putString("phone",phone);
                bundle.putString("mobile", mobile);
                bundle.putString("email", email);
                intent.putExtras(bundle);
                startActivity(intent);
                break;
            case 4:
                contentResolver.delete(CONTENT_URI, "_ID=?", new String[]{menuInfo.id+""});
                break;
        }
        return true;
    }
}
```



说明：

- 第 23 行：定义一个 URI。
- 第 28 行：通过 getContentResolver()获取当前应用的 ContentResolver 实例。
- 第 30 行：通过 URI 进行查询，返回一个 Cursor。Query()方法的第 1 个参数为 URI 地址；第 2 个参数为查询的列名；第 3 个参数是查询条件；第 4 个参数为查询条件的参数；第 5 个参数为排序条件。在这里要查询所有的数据，并且不进行排序，所以后面 3 个参数都为 null。
- 第 91 行：根据_ID 号删除数据。

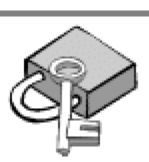
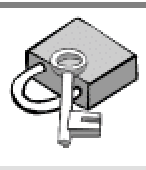
（4）创建 AddPeopleActivity 类文件 AddPeopleActivity.java。在这个 Activity 中，根据从上一个 Activity 中是否有传递数据来判断是修改所传递数据，还是可以向表中插入数据。编写代码如下：

```
package wyq.EX09_4;

import android.app.Activity;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddPeopleActivity extends Activity {
    private EditText edt_name;
    private EditText edt_phone;
    private EditText edt_mobile;
    private EditText edt_email;
    private Button bt_save;
    private ContentResolver contentResolver;
    String name,phone,mobile,email;
    private Uri CONTENT_URI = Uri.parse("content://wyq.EX09_3.Db_People/tb_people");
    Bundle bundle;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.addpeople);
        contentResolver = this.getContentResolver();
        edt_name=(EditText)findViewById(R.id.edt_name);
        edt_phone=(EditText)findViewById(R.id.edt_phone);
        edt_mobile=(EditText)findViewById(R.id.edt_mobile);
        edt_email=(EditText)findViewById(R.id.edt_email);
        bt_save=(Button)findViewById(R.id.bt_save);

```

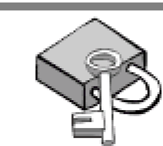





Note

```
        bundle=this.getIntent().getExtras();
        if(bundle!=null)
        {
            edt_name.setText(bundle.getString("name"));
            edt_phone.setText(bundle.getString("phone"));
            edt_mobile.setText(bundle.getString("mobile"));
            edt_email.setText(bundle.getString("email"));
        }
        bt_save.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v) {
            // TODO Auto-generated method stub
            name=edt_name.getText().toString();
            phone=edt_phone.getText().toString();
            mobile=edt_mobile.getText().toString();
            email=edt_email.getText().toString();

            ContentValues value=new ContentValues();
            value.put("name", name);
            value.put("phone", phone);
            value.put("mobile", mobile);
            value.put("email", email);
            long status;
            if(bundle!=null)
            {
              status=contentResolver.update(CONTENT_URI, value, "_ID=?",
                        new String[]{bundle.getLong("id")+""});  ;
            }
            else
            {
              Uri uri2=contentResolver.insert(CONTENT_URI,value);
              if(uri2!=null)
              {
                status=1;
              }
              else
              {
                status=-1;
              }
            }
            if(status!=-1)
            {
              Toast.makeText(AddPeopleActivity.this, "保存成功",
               Toast.LENGTH_LONG).show();
            }
            else
            {
              Toast.makeText(AddPeopleActivity.this, "保存失败",
```



```
                           Toast.LENGTH_LONG).show();
82                   }
83               }
84           });
85       }
86 }
```

说明：

- 第 28 行：获取当前应用的 ContentResolver 实例。
- 第 61 行：根据_ID 号，使用 URI 修改数据。
- 第 65 行：使用 URI 增加数据。

本实例的运行结果与 EX09_3 项目的运行结果相同。

9.5　习　　题

1．使用首选项，对软件进行以下选项设置，如图 9-15 所示。



图 9-15　首选项

2．设计一个简易记事本程序，在该程序中实现以下功能：

（1）查看文本文件。

（2）创建文本文件。

（3）输入文件内容后进行保存。

3．设计一个 Android 程序，进行 XML 文件操作。该程序实现以下功能：

（1）创建 XML 文件。

（2）修改 XML 文件。

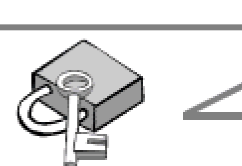
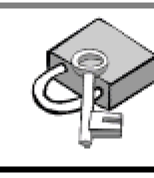
（3）查看 XML 文件。

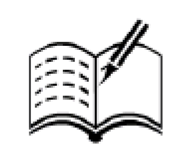
4．设计一个学生成绩信息管理程序，在该程序中实现以下功能：

（1）对学生信息的管理，实现增加、删除、修改、查询操作。

（2）对学生成绩信息的管理，实现增加、删除、修改、查询操作。

学生信息表结构如下：





列　名	含　义	数据类型	说　明
_ID	序号	整型	主键，自增列
StuNO	学号	字符类型	
StuName	学生姓名	字符类型	
Sex	性别	字符类型	
SBirthday	出生日期	日期时间类型	

学生成绩信息表结构如下：

列　名	含　义	数据类型	说　明
_ID	序号	整型	主键，自增列
StuNO	学号	字符类型	
CourseName	课程名	字符类型	
Grade	成绩	整型	

5．编写类文件 StudentGradeProvider.java，实现第 4 题中数据库的 ContentProvider 类。

6．设计一个 Android 程序，通过第 5 题的 StudentGradeProvide 类对学生成绩信息进行管理。

第 10 章 网络通信与服务

【本章内容】

- HTTP 通信
- WebView
- 发送 E-mail
- 消息广播
- Service 组件

随着我国 2009 年开始推广 3G 技术，智能手机迅速普及。智能手机的功能越来越丰富，如高速无线网络、视频通话、手机音乐、手机网游等。无线网络的发展速度也非常快，有了高速无线网络的支持，人们可以随时随地进行数据交换、发送 E-mail、浏览 Internet、浏览微博、手机网上购物等。在 Android 中，掌握网络通信技术便可以开发出优秀的网络应用软件。

Android 除了优秀的界面设计和强大的数据管理功能之外，在网络通信方面也非常优秀，因为 Android 基于 Linux 内核，包含一组优秀的联网功能，并且提供多个网络接口可以使用。可以通过 Android 自带的浏览器来访问网页，也可以通过自带的电子邮件系统来收取邮件。

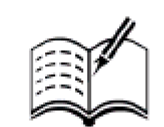
在很多应用程序中，都会通过广播形式发送和接收消息。当操作系统中产生事件时，可以产生一个广播。例如，收到一条短信就会产生一个收到短信的事件，而一旦 Android 操作系统内部产生了这些事件，就会向所有的广播接收器对象来广播这些事件。BroadcastReceiver（广播接收器）是为了实现系统广播而提供的一种组件，并且广播事件处理机制是系统级别的。例如，可以发出一种广播来测试是否收到短信，这时就可以定义一个 BroadcastReceiver 来接收广播，当收到短信时提示用户。当应用程序接收到消息后，一般会启动一个 Activity 或者一个 Service 进行处理。

Service 是在一段不定的时间运行在后台，不和用户交互的应用组件。当应用程序不需要和用户进行交互，或者要占用前台很长时间时，则可以放到后台进行。例如，播放音乐、从 Internet 下载文件等。本章将介绍 Android 平台下网络通信的开发与广播、服务组件的使用。

10.1 HTTP 通信

10.1.1 Android 平台网络接口

Android 平台提供 3 种网络接口可以使用，分别是 java.net.*（标准的 Java 接口）、



org.apache（Apache 接口）、android.net.*（Android 网络接口）。下面分别对这 3 种接口进行介绍。

Note

1. 标准的 Java 接口（java.net.*）

java.net.*提供与联网有关的类，包括流和数据包套接字、Internet 协议及常见的 HTTP 协议。

标准 Java 接口的使用方法如下：

（1）创建 URL。

（2）创建 URLConnection 或者 HttpURLConnection 对象。

（3）设置连接参数。

（4）连接到服务器。

（5）向服务器写数据或者从服务器读取数据。

2. Apache 接口（org.apache）

在 JDK 的 Java.net 包中提供了访问 HTTP 协议的功能，但是对于大部分应用程序来说，其本身的功能还远远不够。在 Android 平台中，引入了 Apache HttpClient，并且进行了一些封装和扩展。可以将 Apache 视为目前流行的开源 Web 服务器。

Apache 接口的使用方法如下：

（1）创建 HttpClient 对象。

（2）创建 Get/Post、HttpRequest 对象。

（3）设置连接参数。

（4）执行 HTTP 操作。

（5）处理服务器返回结果。

3. Android 网络接口（android.net.*）

android.net.*是通过对 Apache 中 HttpClient 的封装来实现的一个 HTTP 编程接口，同时提供了 HTTP 请求队列管理以及 HTTP 连接池管理。除此之外，还有网络状态监视接口、网络访问的 Socket、Uri 类等。

Android 网络接口的使用方法如下：

（1）创建 InetAdress 对象。

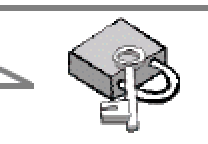
（2）设置 Socket 端口。

（3）获取数据。

（4）处理数据。

10.1.2 HttpClient 接口相关类

HTTP（Hyper Text Transfer Protocol，超文本传输协议）用于传输 WWW 方式的数据，它是一个属于应用层的面向对象的协议，客户端向服务器发送一个请求，服务器以一个状态行作为响应。在 Internet 上，HTTP 通信通常发生在 TCP/IP 连接上，默认端口是 80，但是其他的端口也是可用的。



HTTP 通信中最常用的是通过 GET 和 POST 获取数据。GET 请求可以获取静态页面，也可以把参数放在 URL 后面，传递给服务器。POST 与 GET 的不同之处在于，POST 的参数不是放在 URL 子串后面，而是放在 HTTP 请求数据中。

Android 提供了 HttpURLConnection 和 HttpClient 接口来开发 HTTP 程序。HttpURLConnection 继承于 URLConnection，是 Java 的标准类。Apache 提供了 HttpClient，对 java.net 中的类进行封装，更适合在 Android 上进行联网应用程序的开发，所以本节介绍 HttpClient 接口的使用。

1. HttpGet 类

HttpGet 类是 HTTP 的 GET 方法。GET 方法取回由 Request-URI 标识的以一个实体的形式表示的任何信息。Request-URI 指向产生数据的过程，产生的数据将作为一个实体返回，而不是过程的源文本，除非该文本恰好是该过程的输出所产生的数据。使用该类可以创建一个 HttpGet 连接对象，方法如下：

```
HttpGet httpRequest=new HttpGet(Url);
```

2. HttpPost 类

HttpPost 类是 HTTP 的 POST 方法。POST 方法请求源服务器接收一个请求实体。使用该类可以创建一个 HttpPost 连接对象，方法如下：

```
HttpPost httpRequest=new HttpPost(Url);
```

3. HttpClient 类

HttpClient 类是 HTTP 客户端的一个接口。HTTP 客户端封装了许多对象来执行 HTTP 请求，如处理 Cookie、身份认证、连接管理和其他功能。可以使用 DefaultHttpClient 类创建一个 HTTP 连接，方法如下：

```
HttpClient httpclient=newDefaultHttpClient();
```

4. HttpResponse 类

HttpResponse 类是一个 HTTP 连接响应类。当执行一个 HTTP 连接后，就会返回一个 HttpResponse，可以通过该 HttpResponse 对象获得一些相应信息。例如，获取 HTTP 请求是否成功的方法如下：

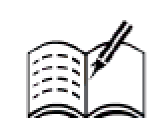
```
HttpResponse httpResponse=httpclient.execute(httpRequest);
if(httpResponse.getStatusLine().getStatusCode()==HttpStatus.SC_OK)
{

}
```

5. ClientConnectionManager 类

ClientConnectionManager 类是客户端连接管理器接口类，提供以下几个方法，如表 10-1

Note


Note

所示。

表 10-1　ClientConnectionManager 提供的方法

方　　法	说　　明
abstract void closeExpiredConnections()	关闭所有无效、超时的连接
abstract void closeIdleConnections(long idletime, TimeUnit tunit)	关闭空闲的连接
abstract void releaseConnection(ManagedClientConnection conn, long validDuration, TimeUnit timeUnit)	释放一个连接
abstract ClientConnectionRequest requestConnection(HttpRoute route, Object state)	请求一个新的连接
abstract void shutdown()	关闭管理器并释放资源

10.1.3　HTTP 通信实例

在 10.1.2 节介绍了 HTTP 通信的相关类，本节将通过一个实例来介绍如何设计 HTTP 通信应用程序。HTTP 请求包括 POST 和 GET 两种方式，本实例将对这两种请求方式的使用进行介绍。

在本例中，首先要架设 Tomcat 服务器，然后通过两种方式获取 Tomcat 服务端的返回信息。本实例的开发步骤如下：

（1）架设 Tomcat 服务器。安装 Tomcat 软件之后，在 Tomcat 的安装目录下的 webapps 目录下新建一个项目文件夹 HttpTest。在文件夹下新建一个文件 httptest.jsp，编写代码如下：

```
1 <%@ page language="java" import="java.util.*" pageEncoding="gb2312"%>
2 <html>
3 <head>
4 <title>Http Test</title>
5 </head>
6 <body>
7 <% String type=request.getParameter("par");
8     String result=new String(type.getBytes("ISO-8859-1"),"gb2312");
9     out.print("<h1>parameters:"+result+"</h1>");
10   %>
11 </body>
12 </html>
```

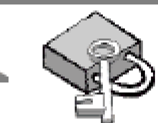
说明：

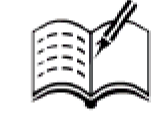
- 第 7 行：获取参数 par 的值。
- 第 8 行：设置字符串的编码。
- 第 9 行：输出显示信息。

（2）新建项目 EX10_1。

（3）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```






```
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10     android:text="这是一个 Http 通信方式的示例"
11     />
12 <Button
13     android:layout_width="fill_parent"
14     android:layout_height="wrap_content"
15     android:text="通过 Get 方式获数据"
16     android:id="@+id/bt_get"
17     />
18 <Button
19     android:layout_width="fill_parent"
20     android:layout_height="wrap_content"
21     android:text="通过 Post 方式获数据"
22     android:id="@+id/bt_post"
23     />
24 <TextView
25     android:layout_width="fill_parent"
26     android:layout_height="fill_parent"
27     android:id="@+id/tv_content"
28     />
29 </LinearLayout>
```

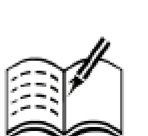
说明：

- 第 2~6 行：声明一个纵向的线性布局，并定义其大小为整个手机屏幕，该布局中包含两个 TextView 控件和两个 Button 控件。
- 第 7~11 行：声明一个 TextView 控件。
- 第 12~17 行：声明一个 ID 为 bt_get 的 Button 控件，用于通过 GET 方式获取数据。
- 第 18~23 行：声明一个 ID 为 bt_post 的 Button 控件，用于通过 POST 方式获取数据。
- 第 24~28 行：声明一个 ID 为 tv_content 的 TextView 控件，用于显示获取的数据。

（4）修改主 Activity 的类文件 FirstActivity.java，实现 HTTP 通信。在本 Activity 中，单击不同的按钮，分别显示 GET 和 POST 的返回信息。编写代码如下：

```
package wyq.EX10_1;

import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
```

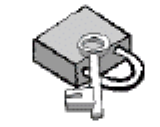

Note

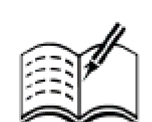
```
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private TextView tv_content;
    private Button bt_get,bt_post;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        tv_content=(TextView)findViewById(R.id.tv_content);
        bt_get=(Button)findViewById(R.id.bt_get);
        bt_post=(Button)findViewById(R.id.bt_post);

        bt_get.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                try
                {
                    final String httpUrl=
                   "http://192.168.1.100:8080/HttpTest/httptest.jsp?par=Get:abcdef";
                    HttpGet httpRequest=new HttpGet(httpUrl);
                    HttpClient httpclient=new DefaultHttpClient();
                    HttpResponse httpResponse=httpclient.execute(httpRequest);
                   if(httpResponse.getStatusLine().getStatusCode()==HttpStatus.SC_OK)
                    {
                        String strResult=EntityUtils.toString(httpResponse.getEntity());
                        tv_content.setText(strResult);
                    }
                    else
                    {
                        tv_content.setText("请求错误");
                    }
                }
                catch(Exception e)
```



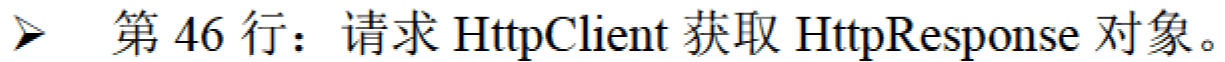


Note

```
58                  {
59                      Toast.makeText(FirstActivity.this, e.getMessage().toString(),
                        Toast.LENGTH_LONG).show();
60                  }
61              }
62          });
63          bt_post.setOnClickListener(new Button.OnClickListener()
64          {
65              @Override
66              public void onClick(View v) {
67          // TODO Auto-generated method stub
68                  try
69                  {
70                      final String httpUrl="http://192.168.1.100:8080/HttpTest/httptest.jsp";
71                      HttpPost httpRequest=new HttpPost(httpUrl);
72                     List <NameValuePair> param=new ArrayList<NameValuePair>();
73                      param.add(new BasicNameValuePair("par","Post Type:abcdef"));
74                      HttpEntity entity=new UrlEncodedFormEntity(param,"utf-8");
75                      httpRequest.setEntity(entity);
76                      HttpClient httpclient=new DefaultHttpClient();
77                      HttpResponse httpResponse=httpclient.execute(httpRequest);
78                      if(httpResponse.getStatusLine().getStatusCode()==HttpStatus.SC_OK)
79                      {
80                          String strResult=EntityUtils.toString(httpResponse.getEntity());
81                          tv_content.setText(strResult);
82                      }
83                      else
84                      {
85                          tv_content.setText("请求错误");
86                      }
87                  }
88                  catch(Exception e)
89                  {
90                      Toast.makeText(FirstActivity.this, e.getMessage().toString(),
                            Toast.LENGTH_LONG).show();
91                  }
92              }
93          });
94      }
95 }
```

说明：

- 第 25、26 行：定义一个 TextView 对象和两个 Button 对象。
- 第 32~34 行：获取 TextView 控件和 Button 控件的引用。
- 第 36~62 行：为 bt_get 按钮控件增加单击监听事件。
  - 第 43 行：定义一个字符串，保存 http 地址。
  - 第 44 行：定义一个 HttpGet 连接对象。
  - 第 45 行：获取 HttpClient 对象。



- ➢ 第 46 行：请求 HttpClient 获取 HttpResponse 对象。
- ➢ 第 47 行：判断请求是否成功，HttpStatus.SC_OK 表示连接成功。
- ➢ 第 49 行：获取返回的字符串。
- ➢ 第 50、54 行：显示返回的字符串信息或者错误信息。
- ➢ 第 59 行：显示 HttpGet 请求的异常信息。

❑ 第 63~93 行：为 bt_post 按钮控件增加单击监听事件。

- ➢ 第 70 行：定义一个字符串，保存 http 地址。
- ➢ 第 71 行：定义一个 HttpPost 连接对象。
- ➢ 第 72 行：使用 NameValuePair 保存要传递的 Post 参数。
- ➢ 第 73 行：添加参数。
- ➢ 第 74 行：设置字符集。
- ➢ 第 75 行：请求 HttpRequest。
- ➢ 第 76 行：获取默认的 HttpClient 对象。
- ➢ 第 77 行：请求 HttpClient 获取 HttpResponse 对象。
- ➢ 第 78 行：判断请求是否成功。
- ➢ 第 80 行：获取返回的字符串。
- ➢ 第 81、85 行：显示返回的字符串信息或者错误信息。
- ➢ 第 90 行：显示 HttpPost 请求的异常信息。

（5）修改配置文件 AndroidManifest.xml。本实例需要访问服务器，需要有访问 Internet 的权限。在 AndroidManifest.xml 文件中加入以下代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

本实例运行结果如图 10-1 和图 10-2 所示。



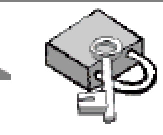
图 10-1　通过 GET 方式获取数据



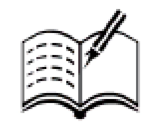
图 10-2　通过 POST 方式获取数据

10.2　WebView

Android 浏览器的内核为 WebKit 引擎。WebKit 引擎是一个开源浏览器网页排版引擎，



具备触摸屏、高级图形显示和上网功能，用户能够在手机上查看邮件、视频节目等。Google 对 WebKit 进行了封装，提供了丰富的 Java 接口，也就是 WebView 控件。在本节将介绍 WebView 控件的相关类及使用。



Note

10.2.1 WebView 类介绍

Android 提供了 WebView 控件进行网页浏览，该控件的使用与其他控件一样，非常方便，在 XML 布局文件中定义一个 WebView 控件，然后在程序中对其各种属性进行设置即可。

1. WebSettings 类

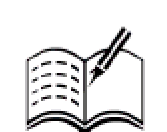
在 Android 中，通过 WebSettings 类对 WebView 的属性、状态等进行设置。WebSettings 对象和 WebView 对象都在同一个生命周期中存在，如果 WebView 对象被销毁或者不存在，使用 WebSettings 则会抛出异常。WebSettings 类常用方法如表 10-2 所示。

表 10-2 WebSettings 类常用方法

方　法	说　明
setAllowFileAccess()	设置启用或禁止 WebView 访问文件数据
setAppCacheEnabled()	设置是否使用缓存
setAppCacheMaxSize()	设置缓存大小
setAppCachePath()	设置缓存路径
setBlockNetworkImage()	设置是否显示网络图像
setBuiltInZoomControls()	设置是否使用内置的视图缩放机制
setCacheMode()	设置缓冲模式
setDatabaseEnabled()	设置使用数据库
setDatabasePath()	设置数据库文件路径
setDefaultFontSize()	设置默认字体大小
setDefaultTextEncodingName()	设置字符默认编码
setJavaScriptEnabled()	设置是否支持 JavaScript
setLayoutAlgorithm()	设置布局方式
setLightTouchEnabled()	设置用鼠标激活被选项
setMinimumFontSize()	设置最小字体
setSaveFormData()	设置保存表单数据
setSavePassword()	设置是否保存密码
setSupportMultipleWindows()	设置是否支持多个窗口
setSupportZoom()	设置 WebView 是否支持缩放
setTextSize()	设置文本大小
setUserAgent()	设置用户代理

将表 10-2 中大部分方法中的 set 改为 get，即可得到 WebView 控件的一些状态和属性。更多的方法参见官方的 API。





2. WebViewClient 类

使用 WebView 调用系统浏览器可以浏览网页。除此之外，可以使用 WebViewClient 在应用程序中自定义网页浏览程序。WebViewClient 类用于辅助 WebView 处理各种通知、请求等事件。可以通过 WebView 的 setWebViewClient()方法来指定一个 WebViewClient 对象。WebViewClient 类常用方法如表 10-3 所示。

表 10-3 WebViewClient 类常用方法

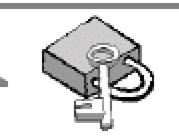
方 法	说 明
doUpdateVisitedHistory()	更新历史记录
onFormResubmission()	应用程序重新请求网页数据
onLoadResource()	加载指定地址提供的资源
onPageFinished()	网页加载完毕
onPageStarted()	网页加载开始
onReceivedError()	报告错误信息
onReceivedHttpAuthRequest()	通知主机应用程序来处理身份验证请求。默认的行为是取消请求
onScaleChanged()	WebView 发生改变
shouldOverrideUrlLoading()	控制新的连接在当前的 WebView 中打开

3. WebChromeClient 类

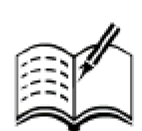
通过 WebViewClient 可以浏览网页的大部分内容。在 Android 中还提供了 WebChromeClient 类，用来帮助 WebView 处理 JavaScript 的对话框、网站图标、加载进度等。WebChromeClient 类常用方法如表 10-4 所示。

表 10-4 WebChromeClient 类常用方法

方 法	说 明
onCloseWindow()	关闭 WebView
onConsoleMessage()	报告一个 JavaScript 控制台消息发送到主机应用程序
onCreateWindow()	创建一个 WebView
onHideCustomView()	通知当前页面的主机应用程序，隐藏自定义视图
onJsAlert()	处理 JavaScript 的 Alert 对话框
onJsBeforeUnload()	告诉客户端显示一个对话框，确认导航离开当前页面
onJsConfirm()	处理 JavaScript 的 Confirm 对话框
onJsPrompt()	处理 JavaScript 的 Prompt 对话框
onJsTimeout()	客户端的 JavaScript 执行时发生超时
onProgressChanged()	加载进度条改变
onReachedMaxAppCacheSize()	程序缓存达到设置的最大值
onReceivedIcon()	网页图标改变
onReceivedTitle()	网页 Title 改变
onRequestFocus()	WebView 显示焦点
onShowCustomView()	通知主机应用程序，当前页面将显示自定义视图



10.2.2　WebView 使用实例



在 10.2.1 节介绍了 WebView 控件的相关类，本节将通过一个实例来介绍 WebView 控件的使用。在本实例中，将模仿制作一个浏览器，实现输入网址，可以对网页进行浏览，并且可以进行前进、后退操作及显示进度条。

本实例开发步骤如下：

（1）创建项目 EX10_2。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<LinearLayout
     android:orientation="horizontal"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     >
     <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/bt_back"
          android:text="后退"
          />
     <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/bt_forward"
          android:text="前进"
          />
     <EditText
          android:layout_width="150px"
          android:layout_height="wrap_content"
          android:id="@+id/ed_url"
          android:singleLine="true"
          android:hint="请输入网址"
          />
     <Button
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:id="@+id/bt_go"
          android:text="GO"
          />
</LinearLayout>
<WebView
```

```
39    android:layout_width="fill_parent"
40    android:layout_height="fill_parent"
41    android:id="@+id/webview"
42    android:focusable="true"
43 />
44 </LinearLayout>
```



说明：

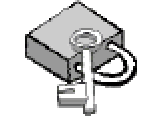
- 第 2~6 行：声明一个纵向的线性布局，并定义其大小为整个手机屏幕，该布局中包含一个嵌套的横向线性布局和一个 WebView 控件。
- 第 7~11 行：声明一个横向的线性布局，该布局包含三个 Button 控件和一个 EditText 控件。
- 第 12~17 行：声明一个 ID 为 bt_back 的 Button 控件，用于实现网页的后退跳转。
- 第 18~23 行：声明一个 ID 为 bt_forward 的 Button 控件，用于实现网页的前进跳转。
- 第 24~30 行：声明一个 ID 为 ed_url 的 EditText 控件，用于输入网址。
- 第 31~36 行：声明一个 ID 为 bt_go 的 Button 控件，用于实现网页的跳转。
- 第 38~43 行：声明一个 ID 为 webview 的 WebView 控件，用于显示网页内容。

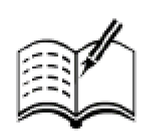
（3）修改主 Activity 的类文件 FirstActivity.java，实现网页的浏览。在本 Activity 中，输入网址后，单击 Go 按钮，实现网页的浏览，并且可以通过“前进”和“后退”按钮实现网页的跳转。编写代码如下：

```
package wyq.EX10_2;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.Window;
import android.webkit.URLUtil;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class FirstActivity extends Activity {
    private WebView webview;
    private Button bt_back,bt_forward,bt_go;
    private EditText ed_url;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

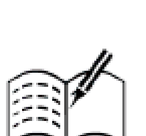




Note

```
        webview=(WebView)findViewById(R.id.webview);
        ed_url=(EditText)findViewById(R.id.ed_url);
        bt_back=(Button)findViewById(R.id.bt_back);
        bt_forward=(Button)findViewById(R.id.bt_forward);
        bt_go=(Button)findViewById(R.id.bt_go);
        WebSettings websettings=webview.getSettings();
        websettings.setBuiltInZoomControls(true);
        websettings.setDefaultFontSize(9);
        websettings.setAllowFileAccess(true);
        webview.setWebViewClient(new WebViewClient(){
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                view.loadUrl(url);
                return true;
            }
        });
        webview.setWebChromeClient(new WebChromeClient()
        {
            @Override
            public void onReceivedTitle(WebView view, String title) {
                FirstActivity.this.setTitle(title);
                super.onReceivedTitle(view, title);
            }
            @Override
            public void onProgressChanged(WebView view, int newProgress) {
                FirstActivity.this.getWindow().setFeatureInt(
                 Window.FEATURE_PROGRESS, newProgress*100);
                super.onProgressChanged(view, newProgress);
            }
        }
        );
        bt_go.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                String url=ed_url.getText().toString().trim();
                if(URLUtil.isNetworkUrl(url))
                {
                    webview.loadUrl(url);
                }
                else
                {
                    Toast.makeText(FirstActivity.this, "网址错误",
                    Toast.LENGTH_LONG).show();
                }
            }

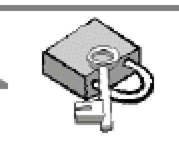
        });
        bt_back.setOnClickListener(new Button.OnClickListener()
```

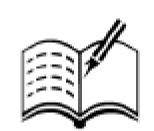

Note

```
72          {
73              @Override
74              public void onClick(View v) {
75                  // TODO Auto-generated method stub
76                  if(webview.canGoBack())
77                  {
78                      webview.goBack();
79                  }
80              }
81
82          });
83          bt_forward.setOnClickListener(new Button.OnClickListener()
84          {
85              @Override
86              public void onClick(View v) {
87                  if(webview.canGoForward())
88                  {
89                      webview.goForward();
90                  }
91              }
92          });
93
94      }
95      public boolean onKeyDown(int keyCode, KeyEvent event)
96      {
97          if ((keyCode == KeyEvent.KEYCODE_BACK) && webview.canGoBack())
98          {
99              webview.goBack();
100             return true;
101         }
102         return false;
103     }
104 }
```

说明：

- 第 18~20 行：定义一个 WebView 对象、三个 Button 对象和一个 EditText 对象。
- 第 26~30 行：获取各个控件的引用。
- 第 31 行：获取 webview 的 WebSettings 对象。
- 第 32 行：设置 webview 支持缩放。
- 第 33 行：设置 webview 的默认字体大小。
- 第 34 行：设置 webview 可以访问文件。
- 第 35~40 行：设置新的连接在当前的 WebView 中打开。
- 第 41~54 行：设置 WebChromeClient。
  - 第 44~46 行：设置应用程序的标题。
  - 第 49~52 行：设置网页加载的进度条。
- 第 55~70 行：为 bt_go 按钮控件添加单击监听事件，实现网页浏览的跳转。





- ➢ 第 59 行：获取 EditText 控件中的内容，并且去除输入内容两边的空格。
- ➢ 第 60 行：判断输入的内容是不是网址。在输入网址时，需要加入“http://”，否则会提示“网址错误”。
- ➢ 第 62 行：为 webview 设置需要访问的网址。

- ❑ 第 71~82 行：为 bt_back 按钮控件添加单击监听事件，实现网页浏览的跳转。
  - ➢ 第 76 行：判断是否能够后退。
  - ➢ 第 78 行：进行后退跳转。
- ❑ 第 83~92 行：为 bt_forward 按钮控件添加单击监听事件，实现网页浏览的跳转。
  - ➢ 第 87 行：判断是否能够前进。
  - ➢ 第 89 行：进行前进跳转。
- ❑ 第 95~103 行：重写 onKeyDown()函数，当按下系统 Back 键，实现后退跳转。如果不做任何处理，浏览网页，按下系统 Back 键，整个 Browser 会调用 finish()而结束自身，如果希望浏览的网页回退而不是退出浏览器，需要在当前 Activity 中处理并重写该 Back 事件。第 97 行判断按下的键是否为系统的 Back 键，并且判断 webview 是否能够后退。

（4）修改配置文件 AndroidManifest.xml。本实例需要访问服务器，需要有访问 Internet 的权限。在 AndroidManifest.xml 文件中加入以下代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

本实例运行结果如图 10-3 和图 10-4 所示。



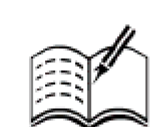
图 10-3　浏览 Google 首页



图 10-4　使用 Google 搜索

10.3　发送 E-mail

在 10.2 节中，介绍了如何通过 WebView 浏览网页。除此之外，很多用户还会通过手机查看电子邮件。本节将介绍在 Android 平台下如何发送 E-mail。


Note

10.3.1 Gmail 简介

通常情况下，发送电子邮件会使用到 SMTP 协议（Simple Mail Transfer Protocel），它是用于由源地址到目的地址传送邮件的规则，通过 SMTP 所指定的服务器便可将 E-mail 发送到收件人的信箱。Android 平台底层便是采用该协议进行通信，而实际上，用户自己开发电子邮件发送程序时通过 Android 内置的 Gmail 程序完成。

在使用 Gmail 程序首发电子邮件时，需要先配置 Gmail 的用户。按照向导需要设置用户的电子信箱、密码、账户类型以及 POP3、SMTP 服务器等信息。在设置成功之后，就可以使用 Gmail 程序收发邮件了，并且在 Gmail 中可以绑定多个账户。

10.3.2 发送 E-mail 实例

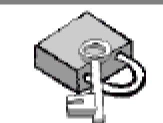
本节将通过一个实例来介绍如何发送 E-mail。方法是调用系统的电子邮件程序，所以首先要配置系统的电子邮件程序。运行程序后不会发送邮件，而是弹出系统电子邮件程序界面，需要单击“发送”按钮后才会发送 E-mail。

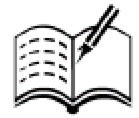
本实例开发步骤如下：

（1）新建项目 EX10_3。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="这是一个发送 Email 的示例"
    />
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="收信人"
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ed_receiver"
        android:hint="输入收信人 Email"
        />
```

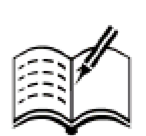



Note

```
27      </LinearLayout>
28 <LinearLayout
29      android:layout_width="fill_parent"
30      android:layout_height="wrap_content"
31      >
32      <TextView
33          android:layout_width="wrap_content"
34          android:layout_height="wrap_content"
35          android:text="主　题"
36          />
37      <EditText
38          android:layout_width="fill_parent"
39          android:layout_height="wrap_content"
40          android:id="@+id/ed_emailSubject"
41          android:hint="输入信件主题"
42          />
43      </LinearLayout>
44 <EditText
45      android:layout width="fill parent"
46      android:layout height="250px"
47      android:id="@+id/ed emailBody"
48      android:hint="输入新建内容"
49      />
50 <Button
51      android:layout_width="fill_parent"
52      android:layout_height="wrap_content"
53      android:id="@+id/bt_send"
54      android:text="发　送"
55      />
56 </LinearLayout>
```

说明：

- 第 2~6 行：声明一个纵向的线性布局，其大小为整个手机屏幕。在该布局中包含一个 TextView 控件、两个嵌套的横向线性布局、一个 EditText 控件和一个 Button 控件。
- 第 12~27 行：声明一个横向的线性布局。该布局嵌套在第一个线性布局中，包含一个 TextView 控件和一个 EditText 控件。第 16~20 行声明一个 TextView 控件；第 21~26 行声明一个 ID 为 ed_receiver 的 EditText 控件，用于输入收信人的电子邮箱地址。
- 第 28~43 行：声明一个横向的线性布局。该布局嵌套在第一个线性布局中，包含一个 TextView 控件和一个 EditText 控件。第 32~36 行声明一个 TextView 控件；第 37~42 行声明一个 ID 为 ed_emailSubject 的 EditText 控件，用于输入 E-mail 主题。
- 第 44~49 行：声明一个 ID 为 ed_emailBody 的 TextView 控件，用于输入 E-mail 信件内容。
- 第 50~55 行：声明一个 ID 为 bt_send 的 Button 控件，单击该控件，发送 E-mail。

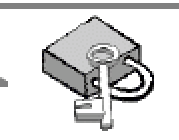

Note

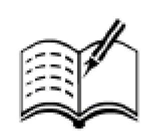
（3）修改主 Activity 的类文件 FirstActivity.java，实现发送 E-mail。编写代码如下：

```
1 package wyq.EX10_3;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9
10 public class FirstActivity extends Activity {
11     /** Called when the activity is first created. */
12     private EditText ed_receiver,ed_emailSubject,ed_emailBody;
13     private Button bt_send;
14     @Override
15     public void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.main);
18
19         ed_receiver=(EditText)findViewById(R.id.ed_receiver);
20         ed_emailSubject=(EditText)findViewById(R.id.ed_emailSubject);
21         ed_emailBody=(EditText)findViewById(R.id.ed_emailBody);
22         bt_send=(Button)findViewById(R.id.bt_send);
23
24         bt_send.setOnClickListener(new Button.OnClickListener()
25         {
26             @Override
27             public void onClick(View v) {
28                 String[] emailReciver = new String[] { ed_receiver.getText().toString() };
29                 String emailSbuject = ed_emailSubject.getText().toString();
30                 String emailbody = ed_emailBody.getText().toString();
31                 Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
32                 emailIntent.setType("plain/text");
33                 emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, emailReciver);
34                 emailIntent.putExtra(android.content.Intent.EXTRA_CC, "test");
35                 emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, emailSbuject);
36                 emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, emailbody);
37                 startActivity(Intent.createChooser(emailIntent, "mail test"));
38             }
39         });
40     }
41 }
```

说明：

- 第 12~13 行：声明三个 EditText 对象和一个 Button 对象。
- 第 19~22 行：获取 EditText 控件和 Button 控件的引用。
- 第 24~40 行：为 bt_send 按钮控件添加单击监听事件，单击该按钮，发送 E-mail。





- 第 28 行：定义一个字符数组，用于存储收信人地址。
- 第 29 行：定义一个字符串，用于存储邮件主题。
- 第 30 行：定义一个字符串，用于存储邮件内容。
- 第 31 行：创建一个 Intent。发送邮件中使用的 Intent 行为为 android.content.Intent.ACTION_SEND。
- 第 32 行：设置邮件格式。
- 第 33 行：将收信人地址添加到 Intent 中。
- 第 34 行：将邮件副本添加到 Intent 中。
- 第 35 行：将邮件主题添加到 Intent 中。
- 第 36 行：将邮件内容添加到 Intent 中。
- 第 37 行：打开 Gmail 发送电子邮件。

（4）修改配置文件 AndroidManifest.xml。本实例需要访问服务器，需要有访问 Internet 的权限。在 AndroidManifest.xml 文件中加入以下代码：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

由于目前模拟器未内置 Gmail Client 端程序，因此发送 E-mail 程序在送出数据后，模拟器上会显示 No Application can perform this action，但是在手机上则可以发送电子邮件。

本实例运行结果如图 10-5 和图 10-6 所示。



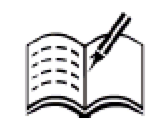
图 10-5　EX10_3 运行结果



图 10-6　Gmail 界面

10.4　消息广播

在 Android 系统中，广播（Broadcast）是在组件之间传播数据（Intent）的一种机制，这些组件甚至可以位于不同的进程中，起到进程间通信的作用。在 Android 系统中，为什么需要广播机制呢？广播机制，其本质上是一种组件间的通信方式，广播的发送者和接收者事先是不需要知道对方的存在的，这样带来的好处便是系统的各个组件可以松耦合地组织在一起，这样系统就具有高度的可扩展性，容易与其他系统进行集成。本节将介绍消息


Note

广播的运行原理和使用。

10.4.1 消息广播运行原理

Android 广播机制包含 3 个基本要素。

- ❑ 广播（Broadcast）：用于发送广播。
- ❑ 广播接收器（BroadcastReceiver）：用于接收广播。
- ❑ 意图内容（Intent）：用于保存广播相关信息的媒介。

BroadcastReceiver 类是对广播消息过滤并响应的类，其运行原理非常简单，即应用程序注册 BroadcastReceiver 之后，当系统或者其他应用程序发送出广播时，所有已经注册的 BroadcastReceiver 会检查注册时的 IntentFilter 是否与发送的 Intent 匹配，若匹配则调用 BroadcastReceiver 的 OnReceive()方法进行处理。所以在开发与 BroadcastReceiver 相关的程序时，主要实现 OnReceive()方法。

1. 发送广播方式

在 Android 中，发送广播有 3 种方式。

（1）sendBroadcast 方式：主要用来广播无序事件，即所有的接收者在理论上是同时接收到事件、同时执行的，对消息传递的效率而言这是比较好的做法。即所有满足条件的 BroadcastReceiver 都会执行其 OnReceive()方法来处理响应，但若有多个满足条件的 BroadcastReceiver 时，其执行 OnReceive()方法的顺序是不固定的。

（2）sendOrderedBroadcast 方式：用来向系统广播有序事件（Ordered Broadcast），接收者按照在 Manifest.xml 文件中设置的接收顺序依次接收 Intent，顺序执行的，接收的优先级可以在系统配置文件中设置（声明在 intent-filter 元素的 android:priority 属性中），数值越大，优先级别越高，其取值范围为-1000~1000。对于有序广播而言，前面的接收者可以对接收到的广播意图（Intent）进行处理，并将处理结果放置到广播意图中，然后传递给下一个接收者，当然，前面的接收者有权终止广播的进一步传播。如果广播被前面的接收者终止，后面的接收器将再也无法接收到广播。即根据 BroadcastReceiver 注册时 IntentFilter 设置的优先级顺序来执行 OnReceive()方法，而相同优先级的 BroadcastReceiver 执行 OnReceive()方法的顺序是不固定的。

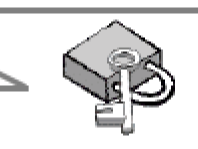
（3）sendStickyBroadcast 方式：与 sendBroadcast 类似，不同之处在于 Intent 在发送之后会一直存在，在以后调用 registerReceiver 注册相匹配的 BroadcastReceiver 时会把该 Intent 直接返回给先注册的 BroadcastReceiver。使用 sendStickyBroadcast 发送广播需要获得 BROADCAST_STICKY permission，如果没有 permission 则会抛出异常。

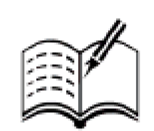
2. 注册 BroadcastReceiver

注册 BroadcastReceiver 有以下两种方法。

（1）静态地在 AndroidManifest.xml 中用<receiver>标签声明注册，并在标签内用<intent-filter>标签注册过滤器。

（2）动态地在代码中先定义并设置一个 IntentFilter 对象，然后在需要注册的地方调





用 Context.registerReceiver()方法，如果取消则调用 Context.unregisterReceiver()方法。如果用动态方式注册的 BroadcastReceiver 的 Context 对象被销毁，BroadcastReceiver 也就自动取消注册。

3．广播接收程序开发过程

广播接收程序的开发过程需要以下几个步骤：

（1）构建 BroadcastReceiver 类的子类，主要重写 OnReceive()方法。

（2）在主程序中发送广播。

（3）为应用程序添加适当的权限。

（4）注册 BroadcastReceive 对象，可以在 AndroidManifest.xml 中静态注册，也可以在类文件中动态注册。

10.4.2　BroadcastReceiver 使用实例

在 10.4.1 节介绍了 BroadcastReceiver 的运行原理及开发过程。本节将通过一个实例来介绍 BroadcastReceiver 的使用方法，其中 BroadcastReceiver 在 AndroidManifest.xml 中静态注册。

本实例开发步骤如下：

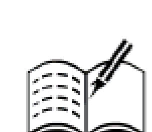
（1）新建项目 EX10_4。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="这是一个 BroadcastReceiver 示例。点击下面的按钮，将会发送一个
       广播。广播接收器中将发送一个 Notification 消息。"
11     />
12 <Button
13     android:layout_width="fill_parent"
14     android:layout_height="wrap_content"
15     android:id="@+id/bt_sendBroad"
16     android:text="发送广播"
17     />
18 </LinearLayout>
```

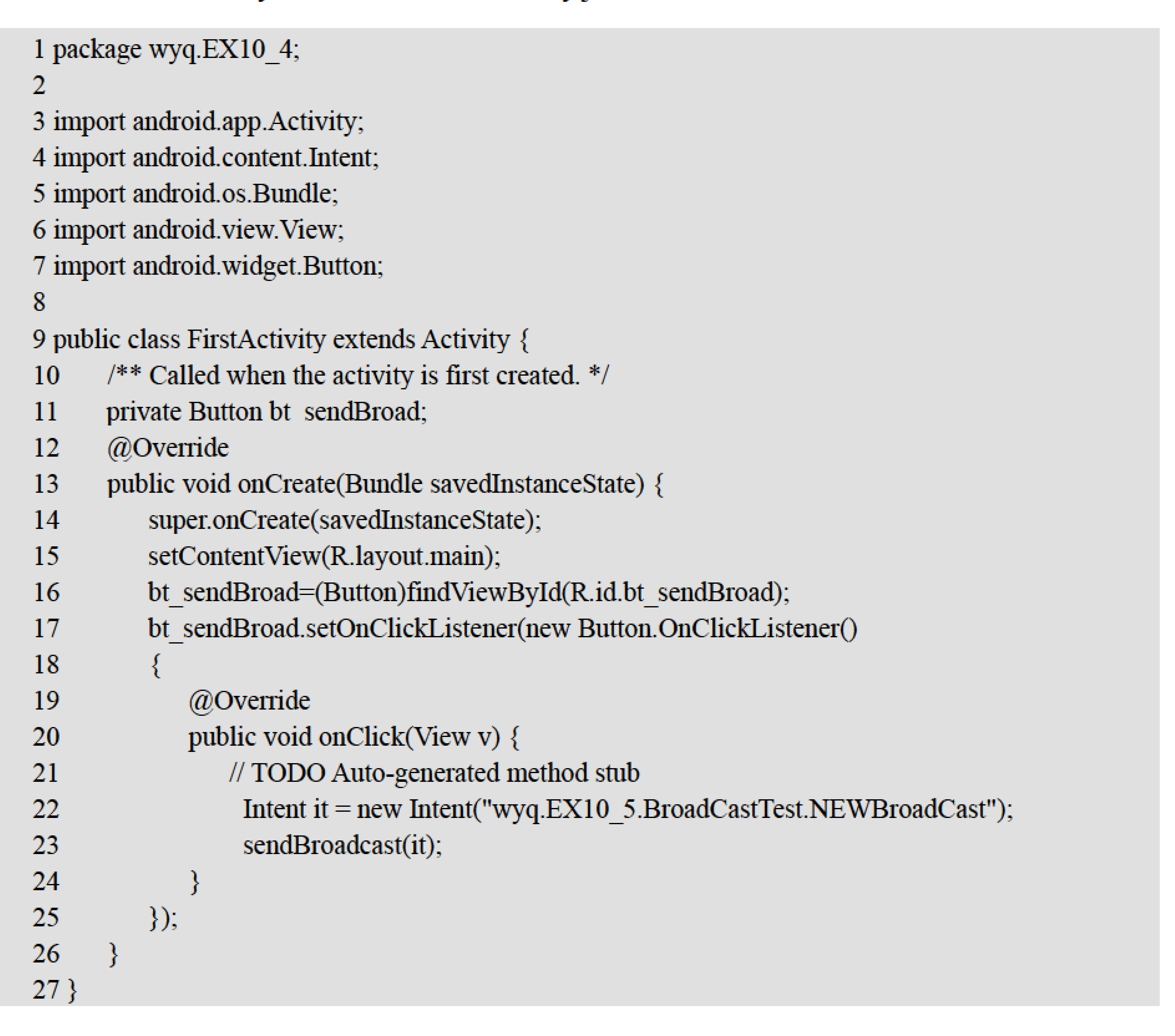
说明：

- 第 2~6 行：声明一个纵向的线性布局，其大小为整个手机屏幕。该布局包含一个 TextView 控件和一个 Button 控件。


Note

- 第 7~11 行：声明一个 TextView 控件，并定义其大小及文本。
- 第 12~17 行：声明一个 ID 为 bt_sendBroad 的 Button 控件，单击该按钮，将发送一个广播。

（3）修改主 Activity 的类文件 FirstActivity.java，实现发送广播。编写代码如下：



```
1 package wyq.EX10_4;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8
9 public class FirstActivity extends Activity {
10      /** Called when the activity is first created. */
11      private Button bt_sendBroad;
12      @Override
13      public void onCreate(Bundle savedInstanceState) {
14          super.onCreate(savedInstanceState);
15          setContentView(R.layout.main);
16          bt_sendBroad=(Button)findViewById(R.id.bt_sendBroad);
17          bt_sendBroad.setOnClickListener(new Button.OnClickListener()
18          {
19              @Override
20              public void onClick(View v) {
21                  // TODO Auto-generated method stub
22                  Intent it = new Intent("wyq.EX10_5.BroadCastTest.NEWBroadCast");
23                  sendBroadcast(it);
24              }
25          });
26      }
27 }
```

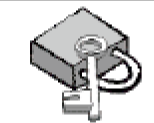
说明：

- 第 11 行：声明一个 Button 类对象。
- 第 16 行：获取 Button 按钮控件的引用。
- 第 17~25 行：为 bt_sendBroad 按钮添加单击监听事件。第 22 行定义一个 Intent 对象，为 BroadcastReceiver 指定 action，使之用于接收相同 action 的广播；第 23 行发送广播。

（4）新建 MyBroadcastReceiver.java 文件，用来接收广播消息。编写代码如下：

```
package wyq.EX10_4;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
```



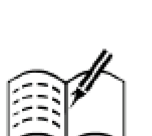

```
7 import android.content.Context;
8 import android.content.Intent;
9
10 public class MyBroadcastReceiver extends BroadcastReceiver {
11     private Notification mNotification = null;
12     private NotificationManager mNotificationManager = null;
13     private Intent mIntent;
14     private PendingIntent mPendingIntent;
15     @Override
16     public void onReceive(Context context, Intent intent) {
17         // TODO Auto-generated method stub
18         mNotificationManager = (NotificationManager)context.getSystemService
            (android.content.Context.NOTIFICATION_SERVICE);
19         mIntent = new Intent(context, FirstActivity.class);
20         mPendingIntent = PendingIntent.getActivity(context, 0, mIntent, 0);
21         mNotification = new Notification();
22         mNotification.icon=R.drawable.icon;
23         mNotification.tickerText = "BroadcastReceiver 测试";
24         mNotification.flags = Notification.FLAG_INSISTENT;
25         mNotification.setLatestEventInfo(context, "BroadcastReceiver 测试",
            "接收 BroadcastReceiver 消息", mPendingIntent);
26         mNotificationManager.notify(1, mNotification);
27     }
28 }
```

说明：

- 第 11 行：声明一个 Notification 对象。
- 第 12 行：声明一个 NotificationManager 对象，用来管理 Notification 对象。
- 第 13 行：声明一个 Intent 对象。
- 第 14 行：声明一个 PendingIntent 对象。
- 第 18 行：通过 getSystemService()方法得到 NotificationManager 对象。
- 第 19 行：定义 Intent 对象，用于启动 SecondActivity 类。
- 第 20 行：定义 PendingIntent 对象，用于跳转到另一个 Activity。
- 第 21 行：定义 Notification 对象。
- 第 22 行：设置 Notification 对象的图标。
- 第 23 行：设置 Notification 对象的提示文字。
- 第 24 行：设置 Notification 对象的 Flag 位。
- 第 25 行：显示在拉伸状态栏中的 Notification 属性，单击后将发送 PendingIntent 对象。
- 第 26 行：提交通知在状态栏中显示。

（5）在 AndroidManifest.xml 文件中注册 BroadcastReceiver，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="wyq.EX10_4"
```


Note

```
4       android:versionCode="1"
5       android:versionName="1.0">
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7         <activity android:name=".FirstActivity"
8                   android:label="@string/app_name">
9             <intent-filter>
10                <action android:name="android.intent.action.MAIN" />
11                <category android:name="android.intent.category.LAUNCHER" />
12            </intent-filter>
13        </activity>
14        <receiver android:name=".MyBroadcastReceiver">
15            <intent-filter>
16                <action android:name="wyq.EX10_5.BroadCastTest.NEWBroadCast" />
17            </intent-filter>
18        </receiver>
19    </application>
20    <uses-sdk android:minSdkVersion="5" />
21 </manifest>
```

说明：

- 第 14~18 行：注册 BroadcastReceiver。
- 第 14 行：android:name=".MyBroadcastReceiver"为处理广播消息的类名。
- 第 16 行：设置广播接收器的过滤事件。

在本例中，当单击 FirstActivity 的 Button 按钮时，将发送 Intent 广播。接收器接收到该广播时，IntentFilter 与发送的 Intent 匹配，则使用 MyBroadcastReceiver 类进行处理，从而发送一个 Notification。

本实例运行结果如图 10-7 和图 10-8 所示。



图 10-7　EX10_4 运行结果

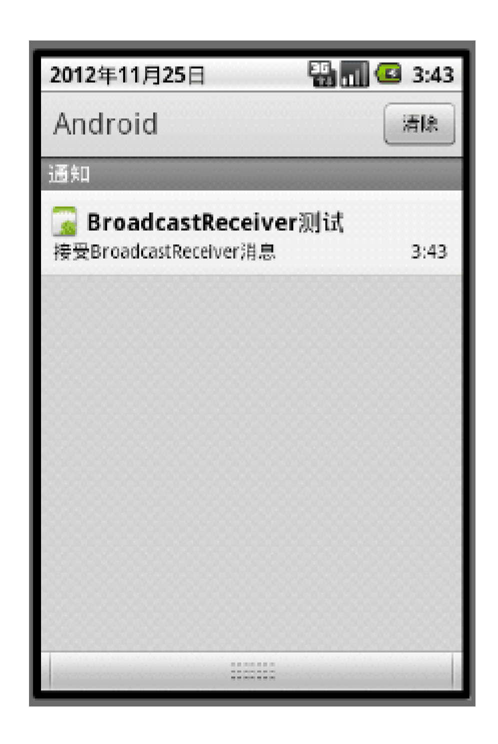
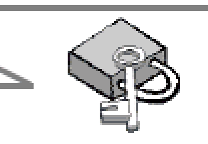


图 10-8　查看通知

10.5　Service 组件

Service（服务）组件是 Android 系统中四个应用程序组件之一，主要用于两个目的：后台运行和跨进程访问。通过启动一个服务，可以在不显示界面的前提下后台运行指定的



任务，这样既可以不占用前台，又可以不影响用户做其他事情。一般使用 Service 为应用程序提供一些服务或不需要界面的功能，例如，从 Internet 下载文件、播放音乐、计时器等。本节主要介绍 Service 的生命周期以及启动 Service 的两种方法，然后通过一个实例来介绍 Service 的使用方法。

Note

10.5.1 Service 的生命周期及启动方法

1. Service 模式及生命周期

Service 有本地服务和远程服务两种模式。

（1）本地服务

本地服务的生命周期不像 Activity 那么复杂，它只继承了 onCreate()、onStart()、onDestroy() 3 个方法。当第一次启动 Service 时，先后调用了 onCreate()、onStart()方法；当停止 Service 时，则执行 onDestroy()方法。这里需要注意的是，如果 Service 已经启动，当再次启动 Service 时，不会再执行 onCreate()方法，而是直接执行 onStart()方法。其生命周期过程为：Context.startService() → onCreate() → onStart() → Service running → 调用 context.stopService()→onDestroy()。

（2）远程服务

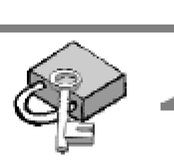
远程服务用于 Android 系统内部的应用程序之间，可以把定义好的接口暴露出来，以便其他应用进行调用操作。客户端建立到服务对象的连接，并通过该连接来调用服务。使用者可以通过调用 Context.bindService() 方法建立连接、启动服务，调用 Context.unbindService()方法关闭连接。多个客户端可以绑定同一个服务，如果服务还没有加载，bindService()会先加载服务。其生命周期过程为：context.bindService()→onCreate()→onBind()→Service running→调用 onUnbind()→onDestroy()。

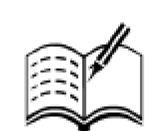
2. Service 启动方法

服务不能自己运行，需要通过调用 startService()或 Context.bindService()方法来启动。这两个方法都可以启动 Service，但是它们的使用场合有所不同。

使用 startService()方法启用服务，调用者与服务之间没有关联，即使调用者退出了，服务仍然运行。如果采用 startService()方法启动服务，在服务没有被创建时，系统会先调用服务的 onCreate()方法，接着调用 onStart()方法。如果调用 startService()方法前服务已经被创建，多次调用 startService()方法并不会导致多次创建服务，但会导致多次调用 onStart()方法。采用 startService()方法启动的服务，只能调用 Context.stopService()方法结束，服务结束时会调用 onDestroy()方法。其过程如图 10-9 所示。

使用 bindService()方法启用服务，调用者与服务绑定在一起，调用者一旦退出，服务也就终止。onBind()只有采用 Context.bindService()方法启动服务时才会回调该方法，该方法在调用者与服务绑定时被调用。当调用者与服务已经绑定，多次调用 Context.bindService()方法并不会导致该方法被多次调用。采用 Context.bindService()方法启动服务时只能调用 onUnbind()方法解除调用者与服务，服务结束时会调用 onDestroy()方法。其过程如图 10-10





所示。

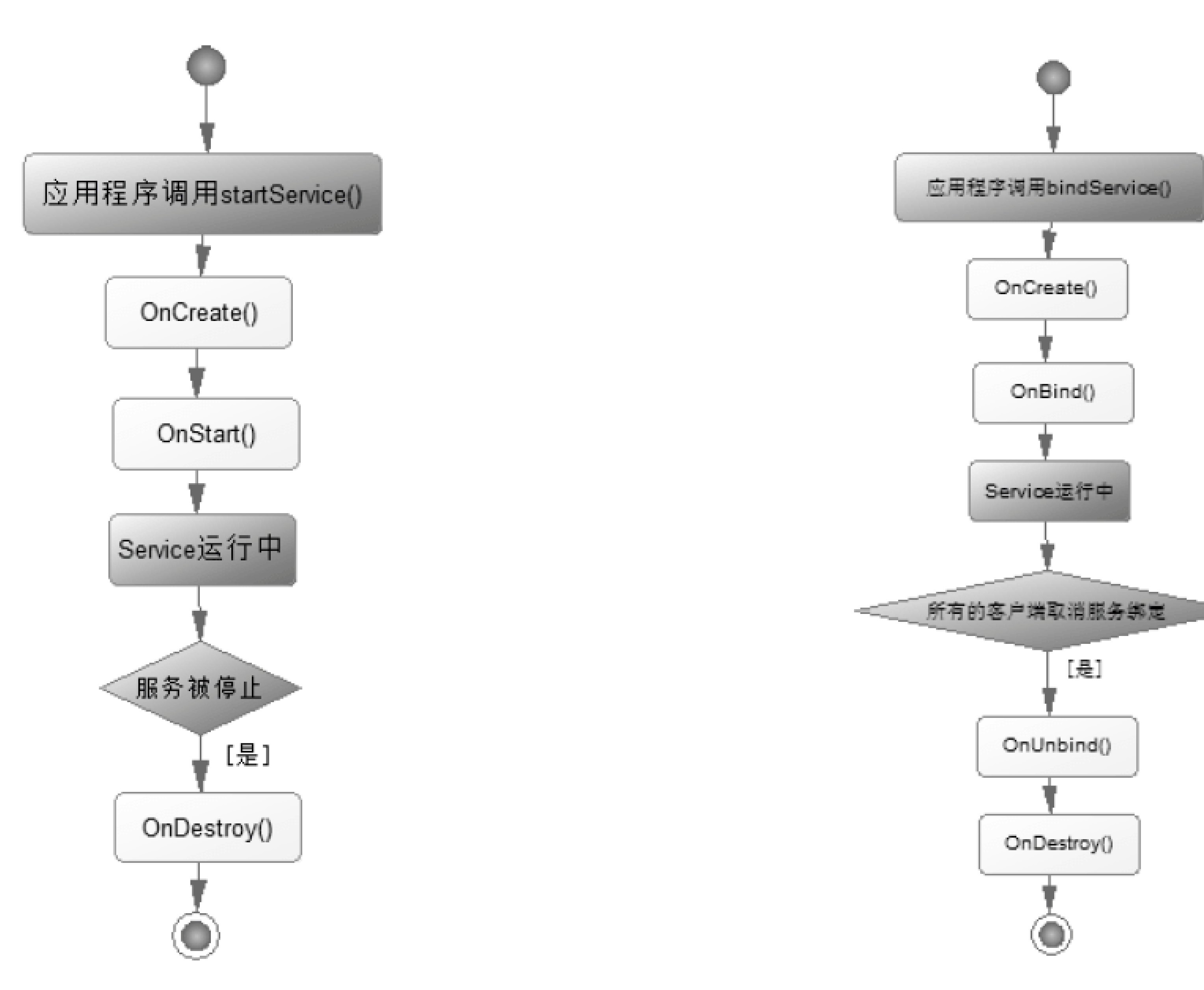


图 10-9　使用 startService()方法启用服务　　　图 10-10　使用 bindService()方法启用服务

10.5.2　Service 使用实例

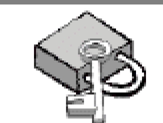
10.5.1 节介绍了 Service 的生命周期及启动方法，本节将通过一个实例来介绍 Service 的使用方法。在本实例中，将介绍 Service 的两种启动方法：OnStart()与 OnBind()。

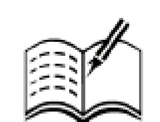
本实例开发步骤如下：

（1）新建项目 EX10_5。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="这是一个 Service 示例"
     />
<Button
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:id="@+id/bt_startService"
     android:text="开始服务"
```






```
17     />
18 <Button
19     android:layout_width="fill_parent"
20     android:layout_height="wrap_content"
21     android:id="@+id/bt_stopService"
22     android:text="停止服务"
23     />
24 <Button
25     android:layout_width="fill_parent"
26     android:layout_height="wrap_content"
27     android:id="@+id/bt_bindService"
28     android:text="绑定服务"
29     />
30 <Button
31     android:layout_width="fill_parent"
32     android:layout_height="wrap_content"
33     android:id="@+id/bt_unBindService"
34     android:text="解除绑定服务"
35     />
36 </LinearLayout>
```

说明：

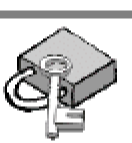
- 第 2~6 行：声明一个纵向的线性布局，其大小为整个手机屏幕。在该布局中包含四个 Button 控件。
- 第 7~11 行：声明一个 TextView 控件。
- 第 12~17 行：声明一个 ID 为 bt_startService 的 Button，单击该按钮开始服务。
- 第 18~23 行：声明一个 ID 为 bt_stopService 的 Button，单击该按钮停止服务。
- 第 24~29 行：声明一个 ID 为 bt_bindService 的 Button，单击该按钮绑定服务。
- 第 30~35 行：声明一个 ID 为 bt_unBindService 的 Button，单击该按钮解除绑定服务。

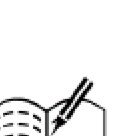
（3）修改主 Activity 的类文件 FirstActivity.java，实现发送广播。编写代码如下：

```
package wyq.EX10_5;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private Button bt_startService,bt_stopService,bt_bindService,bt_unBindService;
```



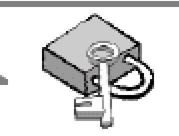


Note

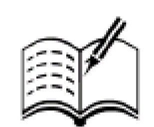
```
private boolean mIsBind;
private MyPlayMusicService mPlayMusicService;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    bt_startService=(Button)findViewById(R.id.bt_startService);
    bt_stopService=(Button)findViewById(R.id.bt_stopService);
    bt_bindService=(Button)findViewById(R.id.bt_bindService);
    bt_unBindService=(Button)findViewById(R.id.bt_unBindService);

    bt_startService.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            startService(new Intent(FirstActivity.this, MyPlayMusicService.class));
        }
    });
    bt_stopService.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            stopService(new Intent(FirstActivity.this, MyPlayMusicService.class));
        }
    });
    bt_bindService.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            bindService(new Intent(FirstActivity.this,MyPlayMusicService.class),
             serviceConnection, Context.BIND_AUTO_CREATE);
            mIsBind = true;
        }
    });
    bt_unBindService.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
              unbindService(serviceConnection);
              mIsBind = false;
        }
    });
}
    private ServiceConnection serviceConnection = new ServiceConnection()
```



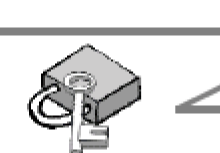
```
65         {
66             @Override
67             public void onServiceDisconnected(ComponentName name)
68             {
69                 mPlayMusicService = null;
70                 Toast.makeText(FirstActivity.this, "Service Failed.",
                   Toast.LENGTH_LONG).show();
71             }
72             @Override
73             public void onServiceConnected(ComponentName name, IBinder service)
74             {
75                 //  获得 MyService 对象
76                 mPlayMusicService = ((MyPlayMusicService.MyBinder)service).getService();
77                 Toast.makeText(FirstActivity.this, "Service Connected.",
                       Toast.LENGTH_LONG).show();
78             }
79         };
80 }
```



Note

说明：

- 第 16 行：声明 4 个 Button 对象。
- 第 17 行：声明一个布尔变量，用于表示服务是否绑定。
- 第 18 行：声明一个 MyPlayMusicService 服务类对象。
- 第 24~27 行：获取 Button 控件的引用。
- 第 29~36 行：为 bt_startService 按钮控件添加单击监听事件。在该事件中，开始服务。第 34 行开始服务。startService()方法的参数是一个 Intent 对象，用于指定 MyPlayMusicService 服务。
- 第 37~44 行：为 bt_stopService 按钮控件添加单击监听事件。在该事件中，停止服务。第 42 行停止服务。stopService()方法的参数是一个 Intent 对象，用于指定 MyPlayMusicService 服务。
- 第 45~53 行：为 bt_bindService 按钮控件添加单击监听事件。在该事件中，绑定服务。第 50 行绑定服务。bindService 方法有 3 个参数，第 1 个参数为 Intent 对象，用于指定 MyPlayMusicService 服务；第 2 个参数为 ServiceConnection 对象，用于连接 Intent 对象指定的服务，通过 ServiceConnection 对象可以获得连接成功或失败的状态，并可以获得连接后的服务对象；第 3 个参数是一个标志位，一般设为 Context.BIND_AUTO_CREATE。
- 第 54~62 行：为 bt_unBindService 按钮控件添加单击监听事件。在该事件中，解除服务绑定。第 59 行解除服务绑定。unbindService()方法的参数为 ServiceConnection 对象。
- 第 64~79 行：声明一个 ServiceConnection 类对象，并重写 onServiceDisconnected()和 onServiceConnected()方法。
    - 第 67~71 行：重写 onServiceDisconnected()方法。当连接服务失败后，该方法被调用。当该方法调用时，使用 Toast 进行提示。





- 第 73~79 行：重写 onServiceConnected()方法。当成功连接服务后，该方法被调用，在该方法中可以获得 MyPlayMusicService 对象，并使用 Toast 进行提示。
- 第 76 行：获取 MyPlayMusicService 对象。

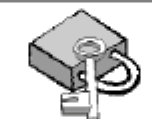
（4）新建 MyPlayMusicService.java 文件，用于构建 MyPlayMusicService 类，该类继承于 Service 类。编写代码如下：

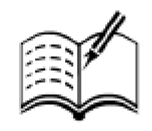
```
package wyq.EX10_5;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Binder;
import android.os.IBinder;
import android.widget.Toast;

public class MyPlayMusicService extends Service {

    MediaPlayer player;
    private MyBinder myBinder = new MyBinder();
    @Override
    public IBinder onBind(Intent arg0) {
        // TODO Auto-generated method stub
        return myBinder;
    }
    @Override
    public void onRebind(Intent intent)
    {
        super.onRebind(intent);
    }
    @Override
    public void onCreate()
    {
         Toast.makeText(this, "My Service Created", Toast.LENGTH_LONG).show();
         player = MediaPlayer.create(this, R.raw.iwear);
         player.setLooping(true);
    }
    @Override
    public void onDestroy()
    {
         Toast.makeText(this, "My Service Stopped", Toast.LENGTH_LONG).show();
         player.stop();
    }
    @Override
    public void onStart(Intent intent, int startid)
    {
         Toast.makeText(this, "My Service Started", Toast.LENGTH_LONG).show();
         player.start();
    }
```






```
43    public class MyBinder extends Binder
44    {
45            MyPlayMusicService getService()
46        {
47            return MyPlayMusicService.this;
48        }
49    }
50 }
```

说明：

- 第 12 行：声明一个 MediaPlayer 类，用于多媒体的播放。
- 第 13 行：声明一个 MyBinder 对象。
- 第 15~18 行：重写 onBind()方法。当服务成功绑定后调用该方法。在该方法中返回 MyBinder 对象。
- 第 20~23 行：重写 onRebind()方法。当服务重新绑定时调用该方法。
- 第 25~30 行：重写 onCreate()方法。当服务创建时调用该方法。在该类中，创建 MediaPlay 对象。第 28 行创建 MediaPlayer 对象，该对象播放 R.raw.iwear 对应的音乐。在运行本实例时，需要在 res 文件夹下建立 raw 文件夹，并放置一个文件名为 iwear.mp3 的音乐文件；第 29 行设置音乐播放模式为循环播放。
- 第 32~36 行：重写 onDestroy()方法。当服务结束时会调用该方法。在该方法中，停止音乐的播放。第 35 行停止播放音乐。
- 第 38~42 行：重写 onStart()方法。当服务开始时会调用该方法。在该方法中，开始播放音乐。第 41 行开始播放音乐。
- 第 43~49 行：定义 MyBinder 类，该类继承于 Binder 类。第 45~48 行定义 MyBinder 类的 getService()方法，在该方法中返回 MyPlayMusicService 对象。

（5）配置 MyPlayMusicService 服务，在 AndroidManifest.xml 文件中增加以下代码：

```
<service android:enabled="true" android:name=".MyPlayMusicService" />
```

本实例运行结果如图 10-11~图 10-15 所示。



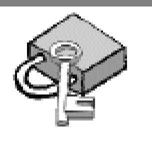
图 10-11　EX10_5 界面



图 10-12　开始服务



图 10-13　停止服务





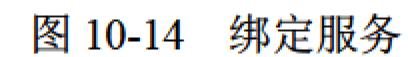


图 10-14　绑定服务



图 10-15　解除绑定服务

10.6　习　　题

1．使用 Tomcat 架设一个 Web 服务器，建立一个个人网站。

2．设计一个 Android 程序，访问在第 1 题中建立的个人网站，通过 GET、POST 两种方式获取信息。

3．设计一个 Android 电子邮件收发程序，实现账户的管理与邮件的接收和发送。

4．设计一个 Android 程序，实现以下功能：当系统时间发生改变时，使用 Alert 对话框进行提示。

5．设计一个音乐播放器程序，使用 Service 实现音乐的后台播放。

Note

第 11 章 手机通信与设置

【本章内容】

- ❑ 拨打电话与电话过滤
- ❑ 收发短信
- ❑ 手机系统设置
- ❑ 手机声音设置
- ❑ 手机闹钟设置

在第 10 章介绍了在 Android 平台中如何通过网络通信进行 Internet 访问、发送电子邮件。Android 平台是一个移动电话平台，除了访问 Internet 之外，还提供拨打和接听电话、收发短信及其他与电话相关的服务。本章将介绍在 Android 平台下，如何进行拨打电话、收发短信等相关的手机通信以及手机的设置。

11.1 拨打电话与电话过滤

手机的基本功能就是拨打和接听电话。对于开发人员，掌握拨打电话技术是非常有用的，根据自己的需求开发一个拨号程序来替换系统自身的拨号程序将是一件非常有意思的事情。本节将介绍如何实现拨打电话和电话过滤。

11.1.1 电话拨号相关类

1. PhoneNumberUtils 类

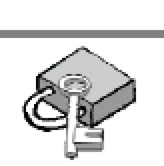
PhoneNumberUtils 类是一个电话号码工具类，用于将字符串数据解析成电话号码。PhoneNumberUtils 类的常用方法如表 11-1 所示。

表 11-1 PhoneNumberUtils 类常用方法

方　法	说　明
formatNumber(String source)	使用默认语言环境设置返回有格式的电话号码
isGlobalPhoneNumber(String phoneNumber)	判断电话号码是否为全球号码

2. TelephonyManager 类

TelephonyManager 类可以获取 Android 电话设备的所有细节信息。在该类中，应用程



序可以确定电话服务和状态，以及访问某些类型的用户信息，还可以注册一个监听器来接收通知的电话状态的变化。TelephonyManager 类的常用方法如表 11-2 所示。

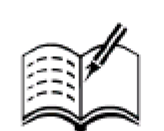

Note


表 11-2　TelephonyManager 类常用方法

方　法	说　明
getCallState()	获取呼叫状态信息
getCellLocation()	获取设备当前位置
getDeviceId()	获取设备 ID
getDeviceSoftwareVersion()	获取设备软件版本
getLine1Number()	获取电话号码
getNetworkType()	获取网络类型
getSimSerialNumber()	获取 SIM 卡序列号
getSimState()	获取 SIM 卡状态
listen(PhoneStateListener listener, int events)	注册一个侦听器对象，用来接收通知中指定的电话状态的变化

3.　PhoneStateListener 类

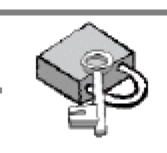
PhoneStateListener 类用于监听电话状态的变化。电话的状态有空闲、通话中和发起呼叫中。在 Android 中可以通过 TelephonyManager 将 PhoneStateListener 实例添加到电话中，这样就可以在状态发生变化时获得通知。PhoneStateListener 类常用的可用状态如表 11-3 所示。

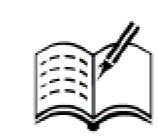
表 11-3　PhoneStateListener 类常用的可用状态

可 用 状 态	说　明
LISTEN_CALL_FORWARDING_INDICATOR	呼叫前指标变化
LISTEN_CALL_STATE	设备通话状态的变化
LISTEN_CELL_LOCATION	设备位置变化
LISTEN_DATA_ACTIVITY	数据流量的方向变化
LISTEN_DATA_CONNECTION_STATE	数据连接状态变化
LISTEN_MESSAGE_WAITING_INDICATOR	消息等待指示变化
LISTEN_NONE	停止监听的更新
LISTEN_SERVICE_STATE	网络服务状态变化
LISTEN_SIGNAL_STRENGTHS	信号强度变化

4.　电话技术相关权限

在开发拨号程序时需要在 AndroidManifest.xml 配置文件中配置相应的权限，才能访问、修改电话状态和拨打电话等。与电话技术相关的常用权限如表 11-4 所示。





Note

表 11-4　电话技术相关权限

权　　限	说　　明
Android.permission. CALL_PHONE	发起电话呼叫
Android.permission. READ_PHONE_STATE	读取电话状态
Android.permission.MODIFY_PHONE_STATE	修改电话状态

11.1.2　拨打电话实例

在 11.1.1 节介绍了拨号程序需要用到的一些类，本节将通过一个实例来介绍在 Android 中如何实现拨号及电话过滤。在本实例中，输入电话号码，单击按钮后进行呼叫。另外，在程序中设置了黑名单，当黑名单上的电话进行呼叫时，会显示提示信息。

本实例的开发步骤如下：

（1）新建项目 EX11_1。

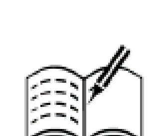
（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

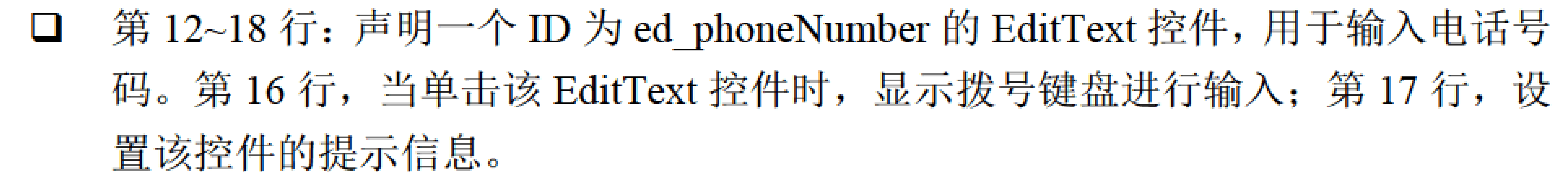
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:text="这是一个拨打电话的示例"
11    />
12 <EditText
13    android:layout_width="fill_parent"
14    android:layout_height="wrap_content"
15    android:id="@+id/ed_phoneNumber"
16    android:inputType="phone"
17    android:hint="请输入电话号码"
18    />
19 <Button
20    android:layout_width="fill_parent"
21    android:layout_height="wrap_content"
22    android:id="@+id/bt_call"
23    android:text="拨号"
24    />
25 </LinearLayout>
```

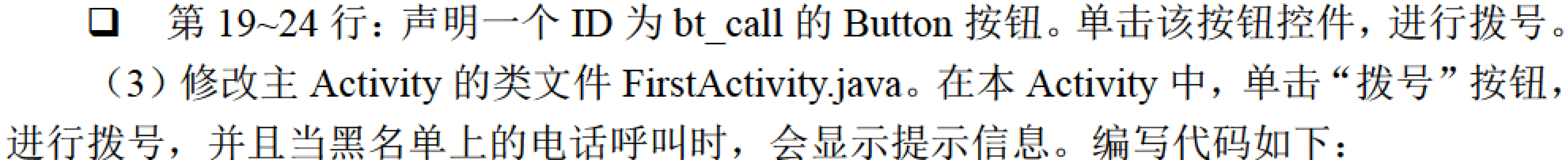
说明：

- 第 2~6 行：声明一个纵向的线性布局，该布局大小为整个手机屏幕。在该布局中包含一个 TextView 控件、一个 EditText 控件与一个 Button 控件。





- 第 7~11 行：声明一个 TextView 控件。

- 第 12~18 行：声明一个 ID 为 ed_phoneNumber 的 EditText 控件，用于输入电话号码。第 16 行，当单击该 EditText 控件时，显示拨号键盘进行输入；第 17 行，设置该控件的提示信息。
- 第 19~24 行：声明一个 ID 为 bt_call 的 Button 按钮。单击该按钮控件，进行拨号。



（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，单击“拨号”按钮，进行拨号，并且当黑名单上的电话呼叫时，会显示提示信息。编写代码如下：

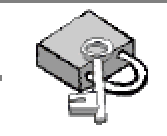
```
package wyq.EX11_1;

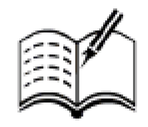
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.PhoneNumberUtils;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private EditText ed_phoneNumber;
    private Button bt_call;
    final String  blackPhoneNumber="5556";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ed_phoneNumber=(EditText)findViewById(R.id.ed_phoneNumber);
        bt_call=(Button)findViewById(R.id.bt_call);

        bt_call.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String phoneNumber=ed_phoneNumber.getText().toString();
                if(PhoneNumberUtils.isGlobalPhoneNumber(phoneNumber))
                {
                    Uri phoneUri=Uri.parse("tel:"+phoneNumber);
                    Intent i=new Intent(Intent.ACTION_CALL,phoneUri);
                    startActivity(i);
                }
                else
```






```
41                  {
42                      Toast.makeText(FirstActivity.this, "电话号码不正确", Toast.LENGTH_SHORT);
43                  }
44              }
45          });
46          MyPhoneStateListener listener=new MyPhoneStateListener();
47          TelephonyManager telManager=
                    (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
48          telManager.listen(listener, MyPhoneStateListener.LISTEN_CALL_STATE);
49      }
50      public class MyPhoneStateListener extends PhoneStateListener
51      {
52      @Override
53          public void onCallStateChanged(int state, String incomingNumber) {
54              // TODO Auto-generated method stub
55              switch(state)
56              {
57              case TelephonyManager.CALL_STATE_IDLE:
58                  Toast.makeText(FirstActivity.this, "电话空闲，可以接听电话",
                        Toast.LENGTH_SHORT).show();
59                  break;
60              case TelephonyManager.CALL_STATE_OFFHOOK:
61                  Toast.makeText(FirstActivity.this, "电话正在通话中",
                        Toast.LENGTH_SHORT).show();
62              case TelephonyManager.CALL_STATE_RINGING:
63                  if(incomingNumber.equals(blackPhoneNumber))
64                      Toast.makeText(FirstActivity.this, "黑名单来电",
                        Toast.LENGTH_SHORT).show();
65                  else
66                      Toast.makeText(FirstActivity.this, "电话空闲，可以接听电话",
                       Toast.LENGTH_SHORT).show();
67                  break;
68              }
69              super.onCallStateChanged(state, incomingNumber);
70          }
71      }
72 }
```

说明：

- 第 17、18 行：声明一个 EditText 对象和一个 Button 对象。
- 第 19 行：定义一个字符串，用来存储黑名单电话。
- 第 25、26 行：获取 EditText 控件和 Button 控件的引用。
- 第 28~45 行：为 bt_call 按钮控件增加单击监听事件，进行拨号。
  - 第 33 行：获取电话号码。
  - 第 34 行：判断电话号码是否合法。
  - 第 36 行：使用 URI 来传入要呼叫的电话号码。注意格式为“tel:电话号码”。

Note

  - ➢ 第 37 行：生成一个 Intent，设置其 Action 为 Intent.ACTION_CALL。
  - ➢ 第 38 行：开始拨号。
- ❑ 第 46 行：声明一个 MyPhoneStateListener 对象，该类继承于 PhoneStateListener，用于监听电话的状态。
- ❑ 第 47 行：通过 getSystemService()方法获取系统服务，获得 TelephonyManager 对象。
- ❑ 第 48 行：添加电话状态的监听。
- ❑ 第 50~71 行：定义 MyPhoneStateListener 类。在该类中，重写 onCallStateChanged()方法，实现电话状态的监听。在该方法中，根据电话的不同状态执行不同操作。当通话状态为来电状态时，判断来电号码是否为黑名单电话，然后显示不同的信息。

（4）修改配置文件 AndroidManifest.xml。本实例需要拨打听话与监听电话状态的权限。在 AndroidManifest.xml 文件中加入以下代码：

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

（5）为了演示本拨号实例，需要再启动一个模拟器，方法是在 dos 输入命令 emulator -data mycall。

本实例运行结果如图 11-1~图 11-3 所示。



图 11-1　EX11_1 界面



图 11-2　拨打电话



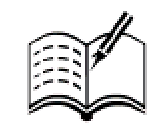
图 11-3　拦截黑名单电话

11.2　收 发 短 信

短信是移动设备上非常重要也是经常使用的一种交流方式，在短信中可以发送简单的文本，也可以将图片包含在里面，发送彩信。在使用 Android 平台的手机中，也内置了短信的应用程序，允许用户接收和发送短信。就像拨号程序一样，根据自己的需求开发一个收发短信的程序来替换系统内置的程序也是一件非常有趣的事情。本节将介绍收发短信的相关类及如何实现收发短信。







11.2.1　收发短信相关类

1. SmsManager 类

SmsManager 类用于管理短信操作，如发送数据、文本和 PDU 短信，通过调用静态方法 SmsManager.getDefault()获取该对象。该类的常用方法如表 11-5 所示。

表 11-5　SmsManager 类常用方法

方　法	说　明
sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)	发送数据短信到一个特定的应用程序端口
sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList <PendingIntent> deliveryIntents)	发送多部分基于文本的 SMS
sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)	发送基于文本的 SMS

2. SmsMessage 类

SmsMessage 类是一个短消息服务信息类。该类的常用方法如表 11-6 所示。

表 11-6　SmsMessage 类常用方法

方　法	说　明
createFromPdu(byte[] pdu)	为原始的 PDU 创建一个 SmsMessage 对象
calculateLength(CharSequence msgBody, boolean use7bitOnly)	计算消息正文和号码所需要的字符数
getServiceCenterAddress()	获取短信服务中心的地址
getOriginatingAddress()	获取信息发出的地址
getMessageBody()	获取消息正文
getDisplayMessageBody()	返回邮件正文或从电子邮件网关发送的电子邮件消息
getDisplayOriginatingAddress()	返回源地址或电子邮件地址
getStatus()	返回短信状态

3. 短信技术相关权限

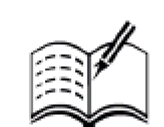
在开发收发短信程序时需要在 AndroidManifest.xml 配置文件中配置相应的权限。与短信技术相关的常用权限如表 11-7 所示。

表 11-7　短信技术相关权限

权　限	说　明
Android.permission. RECEIVE_SMS	监控接收到的短信
Android.permission. READ_SMS	读取短信
Android.permission.SEND_SMS	发送短信
Android.permission.WRITE_SMS	将 SMS 消息写入内置的 SMS 提供程序



11.2.2　收发短信实例



Note

在 11.2.1 节介绍了收发短信需要用到的一些类，本节将通过一个实例来介绍在 Android 中如何收发短信。在本实例中，输入电话号码，单击按钮后发送短信；当收到短信时，会进行提示。

本实例开发步骤如下：

（1）新建项目 EX11_2。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="这是一个收发短信的示例"
11     />
12 <EditText
13     android:layout_width="fill_parent"
14     android:layout_height="wrap_content"
15     android:id="@+id/ed_phoneNumber"
16     android:inputType="phone"
17     android:hint="请输入电话号码"
18     />
19 <EditText
20     android:layout_width="fill_parent"
21     android:layout_height="wrap_content"
22     android:id="@+id/ed_smsText"
23     android:hint="请输入短消息内容"
24     />
25 <Button
26     android:layout_width="fill_parent"
27     android:layout_height="wrap_content"
28     android:id="@+id/bt_sendSMS"
29     android:text="发送短信"
30     />
31 </LinearLayout>
```

说明：

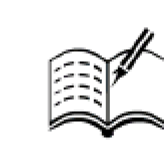
- 第 2~6 行：声明一个纵向的线性布局，其大小为整个手机屏幕。该布局包含一个 TextView 控件、两个 EditText 控件和一个 Button 控件。
- 第 7~11 行：声明一个 TextView 控件。
- 第 12~18 行：声明一个 ID 为 ed_phoneNumber 的 EditText 控件，用于输入接收短



信的电话号码。

- 第 19~24 行：声明一个 ID 为 ed_smsText 的 EditText 控件，用于输入短信内容。
- 第 25~30 行：声明一个 bt_sendSMS 的 Button 控件，单击该按钮，发送短信。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，单击“发送短信”按钮，将发送短信。发送成功后，进行提示。编写代码如下：



```
package wyq.EX11_2;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.PhoneNumberUtils;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class FirstActivity extends Activity {
    /** Called when the activity is first created. */
    private EditText ed_phoneNumber,ed_smsText;
    private Button bt_sendSMS;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ed_phoneNumber=(EditText)findViewById(R.id.ed_phoneNumber);
        ed_smsText=(EditText)findViewById(R.id.ed_smsText);
        bt_sendSMS=(Button)findViewById(R.id.bt_sendSMS);
        final PendingIntent sentIntent=PendingIntent.getActivity(this, 0,
         new Intent(this,FirstActivity.class), 0);
        final SmsManager sManager=SmsManager.getDefault();
        bt_sendSMS.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String phoneNumber=ed_phoneNumber.getText().toString();
                String smsText=ed_smsText.getText().toString();
                if(PhoneNumberUtils.isGlobalPhoneNumber(phoneNumber))
                {
                    sManager.sendTextMessage(phoneNumber, null, smsText, sentIntent, null);
                    Toast.makeText(FirstActivity.this, "短信发送成功",
                     Toast.LENGTH_SHORT).show();
                }
                else
```

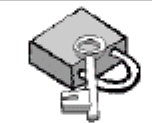
```
41                 {
42                     Toast.makeText(FirstActivity.this, "电话号码不正确",
                        Toast.LENGTH_SHORT).show();
43                 }
44             }
45         });
46     }
47 }
```

说明：

- 第 16、17 行：声明两个 EditText 对象和一个 Button 对象。
- 第 23~25 行：获取 EditText 控件与 Button 控件的引用。
- 第 26 行：定义一个 PendingIntent 对象，作为发送短信的参数。
- 第 27 行：通过 getDefault()方法获取 SmsManager 实例。
- 第 33 行：获取接收短信的电话号码。
- 第 34 行：获取短信内容。
- 第 35 行：判断接收短信的电话号码是否有效。
- 第 37 行：发送短信。

（4）新建 MySmsReceiverListener.java 文件，用来接收短信广播。编写代码如下：

```
1 package wyq.EX11_2;
2
3 import android.content.BroadcastReceiver;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.telephony.SmsMessage;
8 import android.widget.Toast;
9
10 public class MySmsReceiverListener extends BroadcastReceiver {
11
12     @Override
13     public void onReceive(Context context, Intent intent) {
14         // TODO Auto-generated method stub
15         if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED"))
16         {
17             String strMessage="";
18             Bundle bundle=intent.getExtras();
19             if(bundle!=null)
20             {
21                 Object[] pduObjects=(Object[])bundle.get("pdus");
22                 for(Object pduObject :pduObjects)
23                 {
24                     SmsMessage message=SmsMessage.createFromPdu((byte[]) pduObject);
25                     strMessage="收到来自："+message.getOriginatingAddress()+"\n"
                    +message.getMessageBody();
26                 }
```



```
27                  Toast.makeText(context,strMessage, Toast.LENGTH_LONG).show();
28              }
29          }
30      }
31 }
```

说明：

- 第 13 行：过滤 OnReceive()方法中需要的操作，判断接收到的广播是否为短信广播。
- 第 18 行：获取 Intent 中的数据。
- 第 21 行：从 Bundle 中获取短信数据，并放入到 pduObjects 数组中。
- 第 22~25 行：通过循环获取每一条短信。第 24 行创建单条短信；第 25 行构建要显示的字符串。message.getOriginatingAddress()用于获取发送短信的号码，message.getMessageBody()用于获取短信内容。

（5）修改配置文件 AndroidManifest.xml。本实例需要发送和接收短信的权限，并且需要注册接收短信的广播。在 AndroidManifest.xml 文件中加入以下代码：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="wyq.EX11  2"
4       android:versionCode="1"
5       android:versionName="1.0">
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7         <activity android:name=".FirstActivity"
8                   android:label="@string/app_name">
9             <intent-filter>
10                <action android:name="android.intent.action.MAIN" />
11                <category android:name="android.intent.category.LAUNCHER" />
12            </intent-filter>
13        </activity>
14        <receiver android:name=".MySmsReceiverListener">
15            <intent-filter>
16                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
17            </intent-filter>
18        </receiver>
19    </application>
20    <uses-sdk android:minSdkVersion="7" />
21    <uses-permission android:name="android.permission.SEND_SMS"/>
22    <uses-permission android:name="android.permission.RECEIVE  SMS"/>
23 </manifest>
```

说明：

- 第 14~18 行：注册广播接收器。第 14 行设置广播接收器的类名；第 15~18 行设置广播接收器的过滤条件。
- 第 21、22 行：设置程序的权限为发送短信和接收短信。

（6）为了演示本拨号实例，需要再启动一个模拟器，方法是在 dos 中输入命令 emulator


Note

-data mycall。

本实例运行结果如图 11-4~图 11-6 所示。



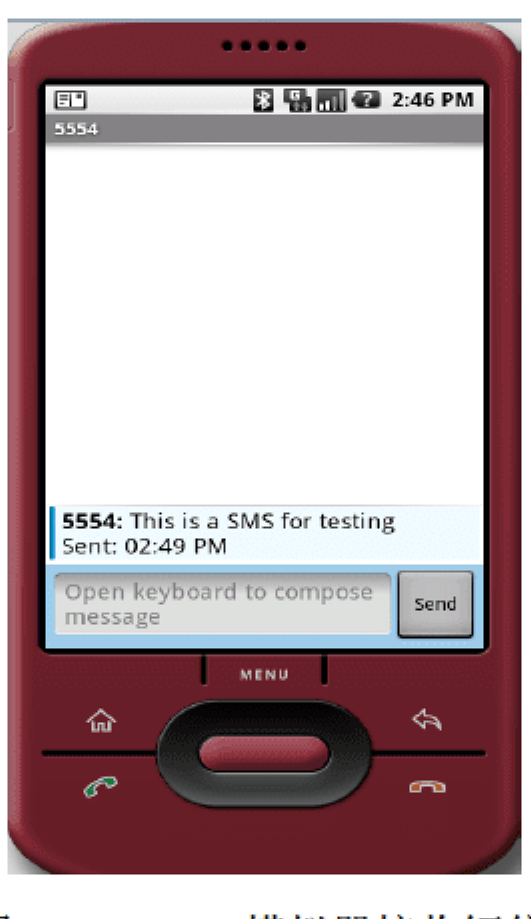





图 11-4　EX11_2 界面　　图 11-5　5554 模拟器接收短信　　图 11-6　模拟器接收短信

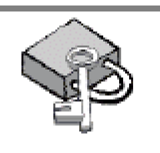
11.3　手机系统设置

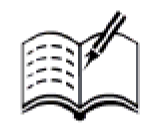
在进行 Android 应用程序开发的过程中，经常需要对系统进行设置，如屏幕方向、震动等。在 Android 的 SDK 中，除了移动网络数据（APN）之外，常用的一些系统设置都有公开的 API 函数。Android 的应用程序开发人员可以通过调用这些 API 函数很方便地对系统进行设置。

在进行系统设置时，需要一定的权限。一些常用的权限如表 11-8 所示。

表 11-8　Android 手机系统常用权限

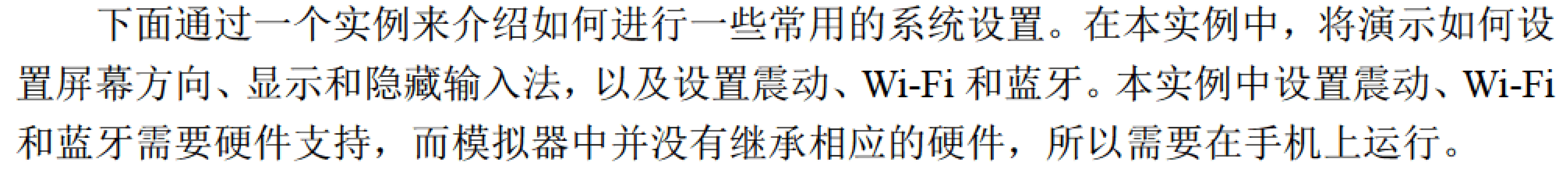
权　　限	说　　明
android.permission.ACCESS_NETWORK_STATE	获取网络信息状态
android.permission.ACCESS_WIFI_STATE	获取当前 Wi-Fi 接入的状态以及 WLAN 热点的信息
android.permission.BATTERY_STATS	获取电池电量统计信息
android.permission.BLUETOOTH	允许程序连接配对过的蓝牙设备
android.permission.BLUETOOTH_ADMIN	蓝牙管理
android.permission.BROADCAST_SMS	当收到短信时触发一个广播
android.permission.CALL_PHONE	允许程序从非系统拨号器里输入电话号码
android.permission.CHANGE_CONFIGURATION	允许当前应用改变配置
android.permission.CHANGE_NETWORK_STATE	改变网络状态
android.permission.CHANGE_WIFI_STATE	改变 Wi-Fi 状态
android.permission.DEVICE_POWER	允许访问底层电源管理
android.permission.EXPAND_STATUS_BAR	允许程序扩展或收缩状态栏
android.permission.GET_PACKAGE_SIZE	获取应用的文件大小





续表

权　　限	说　　明
android.permission.INTERNET	访问网络连接，可能产生 GPRS 流量
android.permission.MODIFY_AUDIO_SETTINGS	修改声音设置信息
android.permission.MODIFY_PHONE_STATE	修改电话状态
android.permission.PROCESS_OUTGOING_CALLS	允许程序监视，修改或放弃播出电话
android.permission.READ_CONTACTS	允许应用访问联系人通讯录信息
android.permission.READ_INPUT_STATE	读取输入状态
android.permission.READ_PHONE_STATE	访问电话状态
android.permission.READ_SMS	读取短信内容
android.permission.REBOOT	允许程序重新启动设备
android.permission.RECEIVE_BOOT_COMPLETE	允许程序开机自动运行
android.permission.RECEIVE_MMS	接收彩信
android.permission.RECEIVE_SMS	接收短信
android.permission.SEND_SMS	发送短信
com.android.alarm.permission.SET_ALARM	设置闹铃提醒
android.permission.SET_DEBUG_APP	设置调试程序，一般用于开发
android.permission.SET_ORIENTATION，	设置屏幕方向
android.permission.SET_TIME	设置系统时间
android.permission.SET_TIME_ZONE	设置系统时区
android.permission.SET_WALLPAPER	设置桌面壁纸
android.permission.STATUS_BAR	允许程序打开、关闭、禁用状态栏
android.permission.VIBRATE	允许振动
android.permission.WAKE_LOCK	允许程序在手机屏幕关闭后，后台进程仍然运行
android.permission.WRITE_CONTACTS	写入联系人，但不可读取
android.permission.WRITE_EXTERNAL_STORAGE	允许程序写入外部存储，如 SD 卡上写文件
android.permission.WRITE_SETTINGS	允许读写系统设置项
android.permission.WRITE_SMS	允许编写短信


下面通过一个实例来介绍如何进行一些常用的系统设置。在本实例中，将演示如何设置屏幕方向、显示和隐藏输入法，以及设置震动、Wi-Fi 和蓝牙。本实例中设置震动、Wi-Fi 和蓝牙需要硬件支持，而模拟器中并没有继承相应的硬件，所以需要在手机上运行。

本实例的开发步骤如下：

（1）新建项目 EX11_3。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

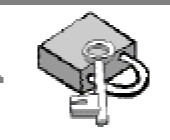
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

Note

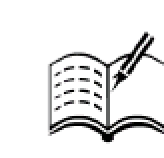
```
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:text="这是一个系统设置更改事件的实例"
11    />
12 <Button
13    android:layout_width="fill_parent"
14    android:layout_height="wrap_content"
15    android:id="@+id/bt_changeScreen"
16    android:text="设置屏幕方向"
17    />
18 <Button
19    android:layout_width="fill_parent"
20    android:layout_height="wrap_content"
21    android:id="@+id/bt_showKey"
22    android:text="显示键盘"
23    />
24 <Button
25    android:layout_width="fill_parent"
26    android:layout_height="wrap_content"
27    android:id="@+id/bt_setVibrator"
28    android:text="设置震动"
29    />
30 <Button
31    android:layout_width="fill_parent"
32    android:layout_height="wrap_content"
33    android:id="@+id/bt_setWifi"
34    android:text="设置 Wifi"
35    />
36 <Button
37    android:layout_width="fill_parent"
38    android:layout_height="wrap_content"
39    android:id="@+id/bt_setBluetooth"
40    android:text="设置蓝牙"
41    />
42 </LinearLayout>
```

说明：

- 第 2~6 行：声明一个纵向的线性布局，其大小为整个手机屏幕。在该布局中包含一个 TextView 控件和 5 个 Button 控件。
- 第 7~11 行：声明一个 TextView 控件。
- 第 12~17 行：声明一个 ID 为 bt_changeScreen 的 Button 控件。单击该按钮，设置屏幕方向。
- 第 18~23 行：声明一个 ID 为 bt_showKey 的 Button 控件。单击该按钮，设置键盘的显示与隐藏。
- 第 24~29 行：声明一个 ID 为 bt_setVibrator 的 Button 控件。单击该按钮，设置手机震动。



- 第 30~35 行：声明一个 ID 为 bt_setWifi 的 Button 控件。单击该按钮，设置 Wi-Fi 的打开与关闭。
- 第 36~41 行：声明一个 ID 为 bt_setBluetooth 的 Button 控件。单击该按钮，设置蓝牙的打开与关闭。



Note

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，可以对手机进行屏幕方向、震动、WiFi 和蓝牙等设置。编写代码如下：

```
package wyq.EX11_3;

import android.app.Activity;
import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.pm.ActivityInfo;

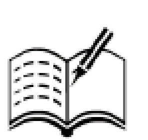
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.Display;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;

public class FirstActivity extends Activity {

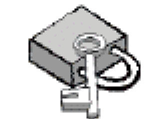
    /** Called when the activity is first created. */
    private Button bt_changeScreen,bt_showKey,bt_setVibrator;
    private Button bt_setWifi,bt_setBluetooth;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

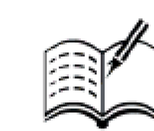
        bt_changeScreen=(Button)findViewById(R.id.bt_changeScreen);
        bt_showKey=(Button)findViewById(R.id.bt_showKey);
        bt_setVibrator=(Button)findViewById(R.id.bt_setVibrator);
        bt_setWifi=(Button)findViewById(R.id.bt_setWifi);
        bt_setBluetooth=(Button)findViewById(R.id.bt_setBluetooth);

        bt_changeScreen.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Display dm = getWindow().getWindowManager().getDefaultDisplay();
                int width = dm.getWidth();
                int height = dm.getHeight();
                if(width>height)
```


Note

```
                {
                 setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                }
                else
                {
                    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                }
            }
        });
        bt_showKey.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                InputMethodManager imm = (InputMethodManager)getSystemService
               (Context.INPUT_METHOD_SERVICE);
                imm.toggleSoftInputFromWindow(v.getWindowToken(), 0,
                 InputMethodManager.HIDE_NOT_ALWAYS);
            }
        });
        bt_setVibrator.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
             Vibrator vibrator=(Vibrator)getSystemService(Service.VIBRATOR_SERVICE);
                vibrator.vibrate(2000);
            }
        });
        bt_setWifi.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                  WifiManager mWifiManager = (WifiManager) getSystemService
                   (Context.WIFI_SERVICE);
                  if (!mWifiManager.isWifiEnabled())
                  {
                          mWifiManager.setWifiEnabled(true);
                   }
                  else
                  {
                      mWifiManager.setWifiEnabled(false);
                  }
            }
        });
        bt_setBluetooth.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                BluetoothAdapter mAdapter= BluetoothAdapter.getDefaultAdapter();
                if(!mAdapter.isEnabled())
```





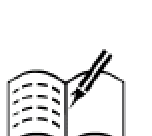

```
88                {
89                    mAdapter.enable();
90                }
91                else
92                {
93                    mAdapter.disable();
94                }
95            }
96        });
97    }
98 }
```

说明：

- 第 20、21 行：声明 5 个 Button 对象。
- 第 27~31 行：获取 Button 控件的引用。
- 第 33~50 行：为 bt_changeScreen 按钮增加单击监听事件，用于改变手机屏幕方向。
  - 第 38 行：获取当前设备对象。
  - 第 39、40 行：获取当前设备的宽度与高度。
  - 第 41~48 行：通过判断宽度与高度的值，来进行屏幕方向设置。
  - 第 43 行：将屏幕方向设置为 PORTRAIT。
  - 第 47 行：将屏幕方向设置为 LANDSCAPE。
- 第 51~58 行：为 bt_showKey 按钮增加单击监听事件，用于显示与隐藏软键盘。
  - 第 55 行：通过 getSystemService 获取 InputMethodManager 对象。
  - 第 56 行：切换软键盘的显示与隐藏。
- 第 59~66 行：为 bt_setVibrator 按钮增加单击监听事件，用于设置震动。
  - 第 63 行：获取 Vibrator 对象。
  - 第 64 行：设置手机震动。
- 第 67~81 行：为 bt_setWifi 按钮增加单击监听事件，用于打开与关闭 Wi-Fi。
  - 第 71 行：获取 WifiManager 对象。
  - 第 72~79 行：判断 Wi-Fi 是否可用，如果不可用，则打开 Wi-Fi；否则关闭 Wi-Fi。
- 第 82~96 行：为 bt_setBluetooth 按钮增加单击监听事件，用于打开与关闭蓝牙。
  - 第 86 行：获取 BluetoothAdapter 对象。
  - 第 87~94 行：判断蓝牙是否使用，如果不可用，则打开蓝牙；否则关闭蓝牙。

（4）修改配置文件 AndroidManifest.xml。本实例对手机进行系统设置。需要设置相应的权限。在 AndroidManifest.xml 文件中加入以下代码：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="wyq.EX11_3"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name" >
        <activity android:name=".FirstActivity"
```


Note

```
8                  android:label="@string/app_name"
9                  android:configChanges="orientation">
10             <intent-filter>
11                 <action android:name="android.intent.action.MAIN" />
12                 <category android:name="android.intent.category.LAUNCHER" />
13             </intent-filter>
14         </activity>
15
16     </application>
17     <uses-sdk android:minSdkVersion="7" />
18    <uses-permission android:name="android.permission.CHANGE_CONFIGURATION"/>
19     <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
20     <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
21     <uses-permission android:name="android.permission.WAKE_LOCK"/>
22     <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
23     <uses-permission android:name="android.permission.BLUETOOTH"/>
24     <uses-permission android:name="android.permission.VIBRATE"/>
25 </manifest>
```

说明：

第 18~24 行：设置本程序需要的权限。权限说明见表 11-8。

本实例的运行结果如图 11-7~图 11-9 所示。



图 11-7　EX11_3 界面



图 11-8　设置屏幕方向



图 11-9　显示输入法

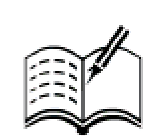
11.4　手机声音设置

在开发 Android 应用程序的过程中，经常会用到对手机声音的设置，如开发音乐播放程序、音乐背景等。本节将介绍在 Android 应用程序中，如何对手机声音进行设置。

11.4.1　AudioManager 类

AudioManager 类是对 Android 进行声音设置的类，在该类中包含了很多对声音模式和



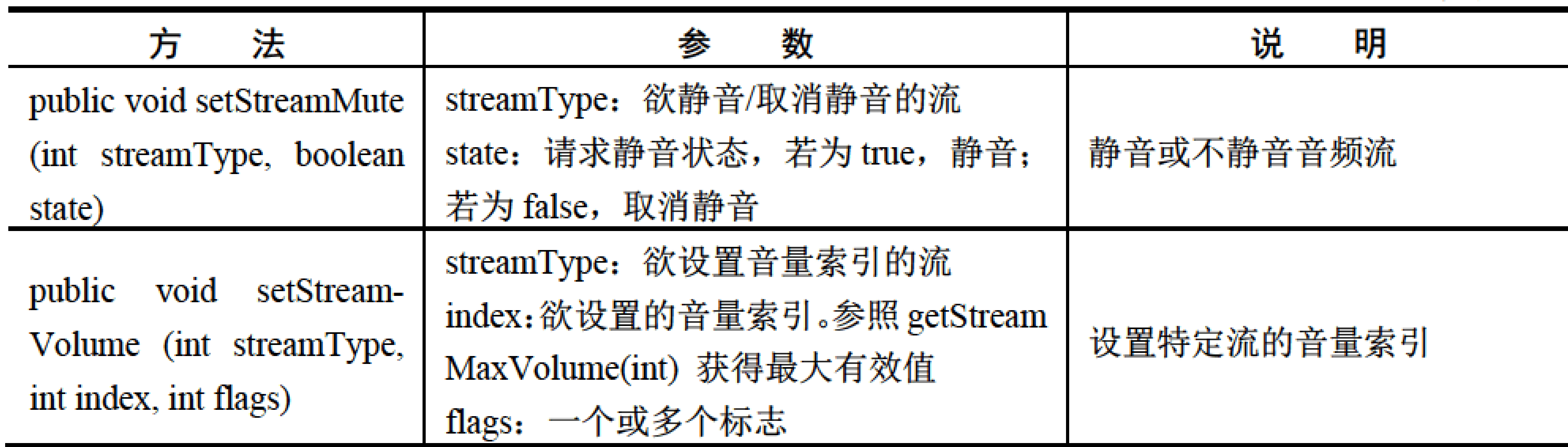


音量进行设置的方法。AudioManager 类对象可以通过 Context 的 getSystemService (Context.AUDIO_SERVICE)来获得。AudioManager 类的常用方法如表 11-9 所示。

表 11-9　AudioManager 类常用方法

<table>
<tr><th>方　法</th><th>参　数</th><th>说　明</th></tr>
<tr><td>public void adjustStream-Volume (int streamType, int direction, int flags)</td><td rowspan="2">streamType：欲调整的流类型。值为 STREAM_VOICE_CALL、STREAM_SY-STEM、STREAM_RING、STREAM_MU-SIC 或 STREAM_ALARM
direction：欲调整音量的方向。值为 ADJUST_LOWER、ADJUST_RAISE 或 ADJUST_SAME
flags：一个或多个标志，值为 FLAG_ALLOW_RINGER_MODES、FLAG_PLA-Y_SOUND、FLAG_REMOVE_ SOUND_AND_VIBRATE 、 FLAG_SHOW_UI 或 FLAG_VIBRATE</td><td>调整指定声音类型的音量</td></tr>
<tr><td>public void adjustVolume (int direction, int flags)</td><td>调整最相关流的音量</td></tr>
<tr><td>public int getMode ()</td><td></td><td>返回当前的音频模式。当前音频模式包括(MODE_NORMAL、MODE_RINGTONE、MODE_IN_CALL 或 MODE_IN_ COMM-UNICATION</td></tr>
<tr><td>public int getRingerMode ()</td><td></td><td>当前铃声模式，值为 RINGER_MODE_NORMAL、RINGER_MODE_SILENT 或 RINGER_MODE_VIBRATE</td></tr>
<tr><td>public int getStreamMax-Volume (int streamType)</td><td rowspan="2">streamType：返回音量索引的流类型</td><td>返回特定流的最大音量索引</td></tr>
<tr><td>public int getStream Volume (int streamType)</td><td>返回特定流的当前音量索引</td></tr>
<tr><td>public boolean isMicrophoneMute ()</td><td></td><td>检查麦克风是否静音</td></tr>
<tr><td>public boolean isSpeakerphoneOn ()</td><td></td><td>检查喇叭扩音器是否打开</td></tr>
<tr><td>public void setMode (int mode)</td><td>mode：请求的音频模式，值为 MODE_NORMAL、MODE_RINGTONE、MOD-E_IN_CALL 或 MODE_IN_COMMUNI-CATION</td><td>设置音频模式</td></tr>
<tr><td>public void setRingerMode (int ringerMode)</td><td>ringerMode：铃声模式。值为 RINGER_MODE_NORMAL、RINGER_MODE_ SI-LENT 或 RINGER_MODE_VIBRATE</td><td>设置铃声模式。静音模式会静音且不会振动；振动模式会静音且会振动；正常模式是有声音且根据用户设置决定是否振动</td></tr>
</table>

续表



方　　法	参　　数	说　　明
public void setStreamMute (int streamType, boolean state)	streamType：欲静音/取消静音的流 state：请求静音状态，若为 true，静音；若为 false，取消静音	静音或不静音音频流
public void setStreamVolume (int streamType, int index, int flags)	streamType：欲设置音量索引的流 index：欲设置的音量索引。参照 getStreamMaxVolume(int) 获得最大有效值 flags：一个或多个标志	设置特定流的音量索引

11.4.2　声音设置实例

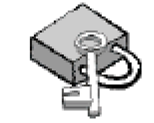
11.4.1 节介绍了设置声音类 AudioManager 的常用方法，本节将通过一个实例来介绍如何在应用程序中设置声音。在本实例中，用户可以设置静音、调整音量大小及铃声模式。

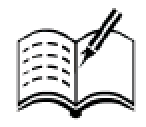
本实例的开发步骤如下：

（1）新建项目 EX11_4。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="这是手机声音设置的实例"
     />
<LinearLayout
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:orientation="horizontal"
     >
     <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/bt_raiseVolume"
          android:text="增大音量"
          />
     <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/bt_lowerVolume"
          android:text="减小音量"
          />
```






```
29     <CheckBox
30         android:layout_width="fill_parent"
31         android:layout_height="wrap_content"
32         android:id="@+id/ck_slient"
33         android:text="静音"
34         />
35 </LinearLayout>
36 <TextView
37     android:layout_width="fill_parent"
38     android:layout_height="wrap_content"
39     android:text="设置铃音模式"
40     />
41 <RadioGroup
42     android:layout_width="fill_parent"
43     android:layout_height="wrap_content"
44     android:orientation="horizontal"
45     >
46     <RadioButton
47         android:layout_width="wrap_content"
48         android:layout_height="wrap_content"
49         android:id="@+id/rb_normal"
50         android:text="正常"
51         />
52     <RadioButton
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:id="@+id/rb_slient"
56         android:text="静音"
57         />
58     <RadioButton
59         android:layout_width="wrap_content"
60         android:layout_height="wrap_content"
61         android:id="@+id/rb_vibrate"
62         android:text="震动"
63         />
64 </RadioGroup>
65 <Button
66     android:layout_width="wrap_content"
67     android:layout_height="wrap_content"
68     android:id="@+id/bt_setRingerMode"
69     android:text="设置"
70     />
71 </LinearLayout>
```

说明：

- 第 2~6 行：声明一个纵向的线性布局，该布局大小为整个手机屏幕。在该布局中包含两个 TextView 控件、一个嵌套的线性布局、一个 RadioGroup 控件和一个 Button 控件。

Note

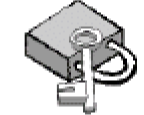
- 第 12~16 行：声明一个横向的线性布局，该布局中包含两个 Button 控件和一个 CheckBox 控件。
- 第 17~22 行：声明一个 ID 为 bt_raiseVolume 的 Button 控件，单击该按钮，可以增大音量。
- 第 23~28 行：声明一个 ID 为 bt_lowerVolume 的 Button 控件，单击该按钮，可以减小音量。
- 第 29~34 行：声明一个 ID 为 ck_slient 的 CheckBox 控件，用于设置是否静音。
- 第 36~40 行：声明一个 TextView 控件。
- 第 41~45 行：声明一个 RadioGroup 控件。在该 RadioGroup 中包含三个 RadioButton 控件。
- 第 46~51 行：声明一个 ID 为 rb_normal 的 RadioButton 控件，表示正常铃声模式。
- 第 52~57 行：声明一个 ID 为 rb_slient 的 RadioButton 控件，表示静音铃声模式。
- 第 58~63 行：声明一个 ID 为 rb_vibrate 的 RadioButton 控件，表示震动铃声模式。
- 第 65~70 行：声明一个 ID 为 bt_setRingerMode 的 Button 控件，单击该按钮，设置铃声模式。

（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，可以调整声音大小及铃声模式。编写代码如下：

```
package wyq.EX11_4;

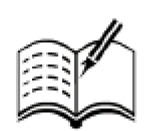
import android.app.Activity;
import android.app.Service;
import android.media.AudioManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.RadioButton;
public class FirstActivity extends Activity {

    private Button bt_raiseVolume,bt_lowerVolume,bt_setRingerMode;
    private RadioButton rb_normal,rb_slient,rb_vibrate;
    private CheckBox ck_slient;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final AudioManager am=(AudioManager)getSystemService (Service.AUDIO_SERVICE);
        bt_raiseVolume=(Button)findViewById(R.id.bt_raiseVolume);
        bt_lowerVolume=(Button)findViewById(R.id.bt_lowerVolume);
        bt_setRingerMode=(Button)findViewById(R.id.bt_setRingerMode);
        rb_normal=(RadioButton)findViewById(R.id.rb_normal);
```

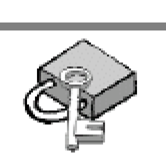


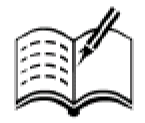
```
        rb_slient=(RadioButton)findViewById(R.id.rb_slient);
        rb_vibrate=(RadioButton)findViewById(R.id.rb_vibrate);
        ck_slient=(CheckBox)findViewById(R.id.ck_slient);

        bt_raiseVolume.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                am.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                 AudioManager.ADJUST_RAISE,AudioManager.FLAG_SHOW_UI);
            }
        });
        bt_lowerVolume.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                am.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                 AudioManager.ADJUST_LOWER,AudioManager.FLAG_SHOW_UI);
            }
        });
        ck_slient.setOnCheckedChangeListener(new OnCheckedChangeListener()
        {
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
                if(ck_slient.isChecked())
                {
                    am.setStreamMute(AudioManager.STREAM_ALARM, true);
                }
                else
                {
                    am.setStreamMute(AudioManager.STREAM_ALARM, false);
                }
            }
        });
        bt_setRingerMode.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                if(rb_normal.isChecked())
                {
                    am.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
                }
                else if(rb_slient.isChecked())
                {
                    am.setRingerMode(AudioManager.RINGER_MODE_SILENT);
                }
                else if(rb_vibrate.isChecked())
                {
                    am.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
```



Note





Note

```
74                }
75            }
76        });
77    }
78 }
```

说明：

- 第 15~17 行：声明 3 个 Button、3 个 RadioButton 和 1 个 CheckBox 对象。
- 第 22 行：通过 getSystemService()获取 AudioManager 对象。
- 第 23~29 行：获取控件的引用。
- 第 31~37 行：为 bt_raiseVolume 按钮增加单击监听事件，用于增大音量。第 35 行增大音量。
- 第 38~44 行：为 bt_ lowerVolume 按钮增加单击监听事件，用于减小音量。第 42 行减小音量，函数参数见表 11-9。
- 第 45~58 行：为 ck_slient 增加监听事件，用于设置是否静音。第 49~56 行根据 ck_slient 是否选中，设置是否静音；第 51 行设置静音；第 55 行取消静音。
- 第 59~76 行：为 bt_setRingerMode 按钮增加单击监听事件，实现设置铃音模式。第 63~74 行根据 RadioButton 控件选中的状态，设置铃音模式；第 65 行设置铃音模式为正常；第 69 行设置铃音模式为静音；第 73 行设置铃音模式为震动。

本实例运行结果如图 11-10 和图 11-11 所示。



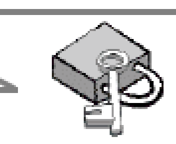
图 11-10　调节音量

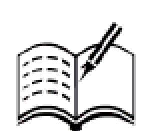


图 11-11　设置铃声模式

11.5　手机闹钟设置

闹钟是生活中不可缺少的重要用品，而闹钟功能也已经成为手机的必备功能之一，所以现在大部分人直接使用手机充当闹钟。在 Android 手机中也内置了闹钟功能，本节将通过一个实例来介绍如何在应用程序中实现手机闹钟的设置。





Note

11.5.1　AlarmManager 类

AlarmManager 类可以在特定的时刻广播一个指定的 Intent。简单地说，就是设定一个时间，然后在该时间到来时，AlarmManager 类会为我们广播一个设定的 Intent。

Android 提供了 4 种类型的闹钟，介绍如下。

（1）ELAPSED_REALTIME：指定的延时后，发送广播，但不唤醒设备。

（2）ELAPSED_REALTIME_WAKEUP：指定的延时后，发送广播，并唤醒设备。

（3）RTC：在指定的时刻发送广播，但不唤醒设备。

（4）RTC_WAKEUP：在指定的时刻发送广播，并唤醒设备。

（5）POWER_OFF_WAKEUP：能唤醒系统，它是一种关机闹钟，即设备在关机状态下也可以唤醒系统。

AlarmManager 类的常用方法如表 11-10 所示。

表 11-10　AlarmManager 类常用方法

方　法	说　明
void cancel(PendingIntent operation)	取消已经注册的与参数匹配的闹钟
void set(int type, long triggerAtTime, PendingIntent operation)	设置一个新的闹钟
void setRepeating(int type, long triggerAtTime, long interval, PendingIntent operation)	设置一个重复类型的闹钟
void setTimeZone(String timeZone)	设置时区。需要 android.permission.SET_TIME_ZONE 权限

11.5.2　手机闹钟设置实例

11.5.1 节介绍了 AlarmManager 的常用方法，本节将通过一个实例来介绍如何在应用程序中设置闹钟。在本实例中，可以设置和取消闹钟，当闹钟时间到后，显示一个 Alert 对话框进行提示。

本实例开发步骤如下：

（1）新建项目 EX11_5。

（2）修改主 Activity 的布局文件 main.xml，编写代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="这是设置手机闹钟的实例"
```





Note

```
11      />
12 <Button
13      android:layout_width="fill_parent"
14      android:layout_height="wrap_content"
15      android:id="@+id/bt_setClock"
16      android:text="设置闹钟"
17      />
18 <Button
19      android:layout_width="fill_parent"
20      android:layout_height="wrap_content"
21      android:id="@+id/bt_cancelClock"
22      android:text="取消闹钟"
23      />
24 <TextView
25      android:layout_width="fill_parent"
26      android:layout_height="wrap_content"
27      android:id="@+id/tv"
28      />
29 </LinearLayout>
```

说明：

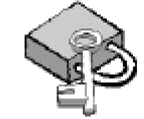
- 第 2~6 行：声明一个纵向的线性布局，该布局大小为整个手机屏幕，包含两个 TextView 控件和两个 Button 控件。
- 第 7~11 行：声明一个 TextView 控件。
- 第 12~17 行：声明一个 ID 为 bt_setClock 的 Button 控件。单击该按钮，设置闹钟。
- 第 18~23 行：声明一个 ID 为 bt_cancelClock 的 Button 控件。单击该按钮，取消闹钟。
- 第 24~28 行：声明一个 ID 为 tv 的 TextView 控件，用于显示闹钟信息。

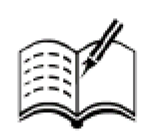
（3）修改主 Activity 的类文件 FirstActivity.java。在本 Activity 中，可以设置和取消闹钟。编写代码如下：

```
package wyq.EX11_5;

import java.util.Calendar;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

public class FirstActivity extends Activity {
     private Button bt_setClock,bt_cancelClock;
```






```
private TextView tv;
AlarmManager am;
Calendar calendar;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    calendar = Calendar.getInstance();
    am = (AlarmManager) getSystemService(ALARM_SERVICE);
    tv=(TextView)findViewById(R.id.tv);
    bt_setClock=(Button)findViewById(R.id.bt_setClock);
    bt  cancelClock=(Button)findViewById(R.id.bt  cancelClock);
    bt_setClock.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            calendar.setTimeInMillis(System.currentTimeMillis());
            new TimePickerDialog(
                    FirstActivity.this,
                    new TimePickerDialog.OnTimeSetListener() {
              public void onTimeSet(TimePicker view,int hourOfDay, int minute)
                        {
                            calendar.setTimeInMillis(System.currentTimeMillis());
                            calendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
                            calendar.set(Calendar.MINUTE, minute);
                            calendar.set(Calendar.SECOND, 0);
                            calendar.set(Calendar.MILLISECOND, 0);
                             Intent intent = new Intent(FirstActivity.this, AlarmReceiver.class);
                             PendingIntent pendingIntent = PendingIntent
                             .getBroadcast(FirstActivity.this, 0,intent, 0);
                            am.setRepeating(AlarmManager.RTC_WAKEUP,
                             System.currentTimeMillis()+ (10 * 1000),
                             (24 * 60 * 60 * 1000),pendingIntent);
                            Toast.makeText(FirstActivity.this, "闹钟设置成功",
                             Toast.LENGTH_LONG).show();
                            tv.setText("设置的闹钟时间是： "+hourOfDay+":"+minute);
                        }
                    },
                    calendar.get(Calendar.HOUR_OF_DAY),
                    calendar.get(Calendar.MINUTE),
                    true).show();
    }
    });
    bt_cancelClock.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View arg0) {
            Intent intent = new Intent(FirstActivity.this, AlarmReceiver.class);
            PendingIntent pendingIntent = PendingIntent. getBroadcast
           (FirstActivity.this, 0, intent, 0);
```



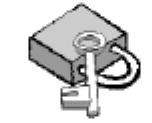
```
64                  am.cancel(pendingIntent);
65                  tv.setText("");
66                  Toast.makeText(FirstActivity.this, "闹钟已取消", Toast.LENGTH_LONG).show();
67              }
68          });
69      }
70 }
```

说明：

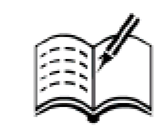
- 第 17、18 行：声明两个 Button 对象和一个 TextView 对象。
- 第 19、20 行：声明一个 AlarmManager 对象和一个 Calendar 对象。
- 第 26 行：获取日期对象。
- 第 27 行：通过 getSystemService()获取 AlarmManager 对象。
- 第 28~30 行：获取 TextView 控件和 Button 控件的引用。
- 第 31~57 行：为 bt_setClock 按钮增加单击监听事件，实现设置闹钟。
  - 第 35 行：设置 Calendar 对象。
  - 第 36 行：创建并显示时间选择对话框。
  - 第 38~55 行：创建 OnTimeSetListener 监听器。
  - 第 42 行：设置闹钟的小时数。
  - 第 43 行：设置闹钟的分钟数。
  - 第 44 行：设置闹钟的秒数。
  - 第 45 行：设置闹钟的毫秒数。
  - 第 46、47 行：建立 Intent 和 PendingIntent 来调用目标组件。
  - 第 48 行：设置重复闹钟，重复频率为每天一次。24 * 60 * 60 * 1000 为一天的毫秒数。
  - 第 49 行：闹钟设置成功后，使用 Toast 进行消息提示。
  - 第 50 行：在 TextView 中显示闹钟信息。
  - 第 53 行：传入当前小时数。
  - 第 54 行：传入当前分钟数。
  - 第 55 行：设置是否为 24 小时制。true 代表以 24 小时制显示时间。
- 第 58~68 行：为 bt_cancelClock 按钮增加单击监听事件，实现取消闹钟。
  - 第 62、63 行：建立 Intent 和 PendingIntent 来调用目标组件。
  - 第 64 行：取消闹钟。
  - 第 65 行：设置 TextView 控件内容。
  - 第 66 行：闹钟取消后，使用 Toast 进行消息提示。

（4）新建广播接收类文件 AlarmReceiver.java。该类继承于 BroadCastReceiver 类，其功能是接收闹钟时间到后被广播的 Intent 对象。编写代码如下：

```
1 package wyq.EX11  5;
2
3 import android.content.BroadcastReceiver;
```



```
4 import android.content.Context;
5 import android.content.Intent;
6
7 public class AlarmReceiver extends BroadcastReceiver {
8     @Override
9     public void onReceive(Context context, Intent intent) {
10         // TODO Auto-generated method stub
11         Intent i=new Intent(context,AlarmActivity.class);
12         i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
13         context.startActivity(i);
14     }
15 }
```



说明：

- 第 11 行：创建 Intent 对象，用于跳转到 AlarmActivity。
- 第 12 行：设置 Intent 的标志。
- 第 13 行：启动 Activity。

（5）新建类文件 AlarmActivity.java，该类用于闹钟时间到后显示用户的提醒界面。编写代码如下：

```
package wyq.EX11_5;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;

public class AlarmActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        new AlertDialog.Builder(AlarmActivity.this)
            .setTitle("闹钟")
            .setMessage("时间到了")
            .setPositiveButton("确定",
                new OnClickListener()
                {
                    @Override
                    public void onClick(DialogInterface dialog, int which)
                    {
                        finish();
                    }
                })
                    .create()
                    .show();
    }
}
```

Note

说明：

第 13~26 行：创建并显示 Alert 对话框。

- 第 14 行：设置对话框标题。
- 第 15 行：设置对话框消息。
- 第 16~24 行：设置对话框的按钮，并增加单击监听事件。单击该按钮后，关闭 AlarmActivity。
- 第 25 行：创建 Alert 对话框。
- 第 26 行：显示 Alert 对话框。

本实例运行结果如图 11-12 和图 11-13 所示。



图 11-12　设置闹钟



图 11-13　闹钟提醒

11.6　习　　题

1．设计一个 Android 程序，实现 Wi-Fi 的打开与关闭。

2．设计一个 Android 短信收发程序，实现以下功能：

（1）实现短信的发送。

（2）对黑名单上电话发送的短信进行过滤。

（3）黑名单可以自行增加和删除。

3．设计一个 Android 闹钟程序，实现以下功能：

（1）闹钟提醒时间设置。

（2）设置闹钟的间隔提醒时间。

（3）设置闹钟的提醒频率。

4．设计一个 Android 拨打电话程序，实现以下功能：

（1）拨打电话。

（2）对黑名单上电话拨打的电话进行过滤。

（3）将电话号码增加到通讯录。

第 12 章

Android 游戏制作

【本章内容】

- ❑ Android 游戏的基础技术
- ❑ 贪吃蛇游戏的解析
- ❑ 贪吃蛇游戏的功能拓展

前面介绍了 Android 系统的常用组件和布局，利用这些知识可以完成一些应用程序的界面设计。本章将介绍一个入门级的小游戏——贪吃蛇。贪吃蛇（Snake）是一个经典游戏，最早应该追溯到诺基亚的黑白机时代，是诺基亚手机上的一款经典小游戏，现在的贪吃蛇游戏出现了许多新的版本，并被借鉴到各种平台上，Android SDK 1.5 上就有其身影，本章通过贪吃蛇游戏实例，介绍 Android 的图形绘制、贴图方法和游戏开发的基本逻辑和设计流程，同时对该游戏加以扩充，增加触摸屏功能、背景图片和背景音乐效果，使其更具实用性。

12.1　Android 游戏的基础技术

12.1.1　Android 的简单图形绘制

Android 绘图操作通过继承 View 实现，在 onDraw()函数中实现绘图。下面通过一个绘制图形的实例，让大家对简单图形的绘制机理有一个更好的了解。

本实例开发步骤如下：

（1）新建项目 EX12_1，其目录结构如图 12-1 所示。

（2）在 src 下创建两个类，一个是继承了 View，用来画图的 MyView 类；另外一个是用来调用创建视图的 MainActivity 类。

① MainActivity 类的源代码如下：

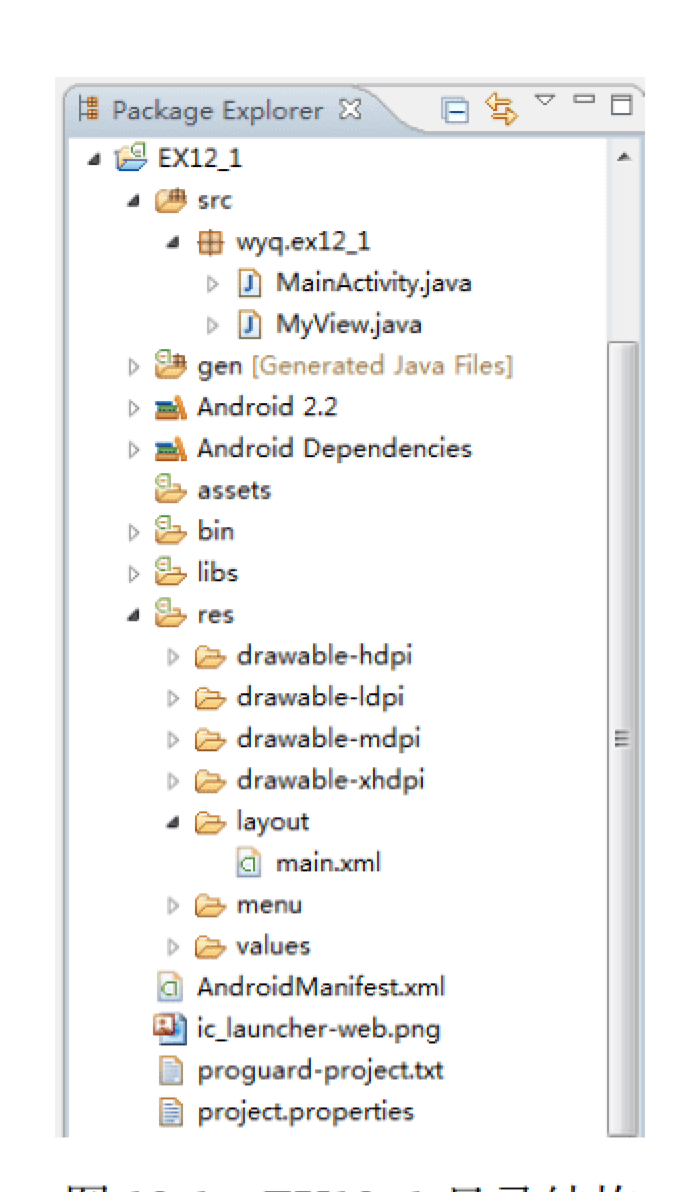


图 12-1　EX12_1 目录结构

Note

```
1  public class MainActivity extends Activity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(new MyView(this));//调用自定义的 MyView
7      }
8  }
```

说明：

第 6 行： Activity 调用自定义的 MyView 类。

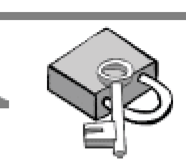
游戏的核心是不断地绘图和刷新界面源，MainActivity 类主要用来调用我们自定义的视图。

② MyView 类的源代码如下：

```
1  public class MyView extends View {
2
3      public MyView(Context context) {                           //Activity 中初始化使用
4          super(context);
5      }
6
7      @Override
8      protected void onDraw(Canvas canvas) {                     //重写的绘制方法
9          super.onDraw(canvas);
10
11         Paint paint = new Paint();                              //创建画刷
12         paint.setAntiAlias(true);                               //设置抗锯齿效果
13         paint.setColor(Color.WHITE);                            //设置画刷的颜色
14
15         paint.setStyle(Paint.Style.FILL);                       //用颜色填充图形
16         canvas.drawColor(Color.BLACK);                          //设置背景颜色
17         canvas.drawRect(10, 10, 100, 100, paint);               //绘制矩形
18         canvas.drawCircle(55, 150, 45, paint);                  //绘制圆
19
20         paint.setStyle(Paint.Style.STROKE);                     //绘制边框
21         canvas.drawRect(110, 10, 200, 100, paint);              //绘制矩形
22         canvas.drawCircle(155, 150, 45, paint);                 //绘制圆形
23
24         canvas.drawText("这是一个字符串", 10, 230, paint);      //绘制字符串
25         canvas.drawLine(120, 230, 200, 230, paint);             //绘制直线
26     }
27 }
```

说明：

- 第 3 行：重写 View 下的构造函数。View 下的构造函数有 3 种，如果在 XML 中配置应用该 View，必须实现该构造函数。
- 第 11 行：Paint 可以理解为画刷或者画笔，主要用来设置绘图使用的颜色、填充



方式、透明度、字体以及字体样式等。

- 第 12 行：用来设置抗锯齿效果。
- 第 16~18 行：Canvas 画布，在 View 上显示的图形都是由 canvas 来绘制的。Canvas 绘图需要用 Paint 来设置颜色和样式。在本例的图形中用到了 Canvas 的 drawRect()、drawCircle()、drawText()和 drawLine()方法分别绘制矩形、圆形、字符串和直线。
- 第 15、20 行：对比了 Paint 的两种样式效果，Paint.Style.FILL 用颜色填充图形；Paint.Style.STROKE 绘制边框。

（3）将 res\layout 下的 main.xml 的代码修改如下：

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       >
7       <!--添加一个自定义的 View 到线性布局中   -->
8       <wyq.ex12_1.MyView
9           android:id="@+id/myView"
10          android:layout_width="match_parent"
11          android:layout_height="match_parent" />
12  </LinearLayout>
```

说明：

第 8~11 行：在垂直的线性布局中添加了一个自定义的 View，并指定了 ID，宽度和高度全部填充父控件。

项目 EX12_1 的运行结果如图 12-2 所示。

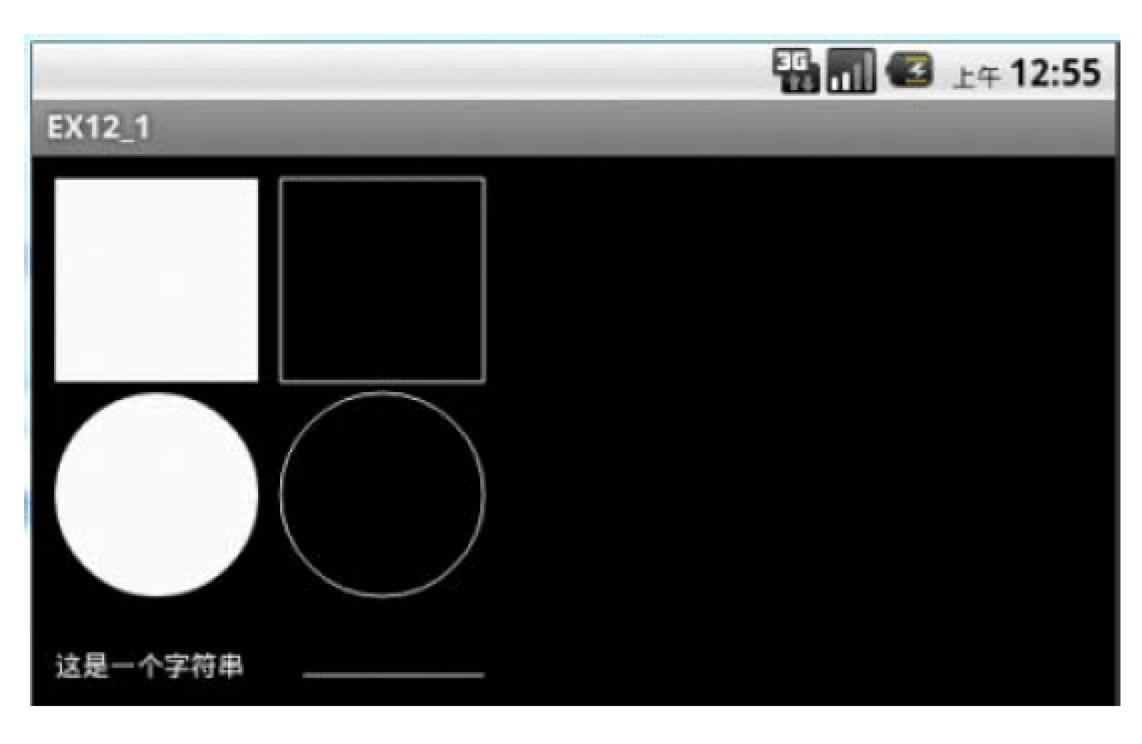
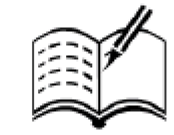


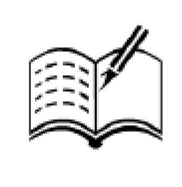
图 12-2　EX12_1 运行结果



Note

12.1.2　Android 的贴图技术

掌握了简单图形的绘制还不能满足开发的需求，毕竟一个漂亮的界面对用户是最有吸引力的。贴图技术主要包括图片的移动、旋转、透明度等变化。位图是开发中最常用的资源，下面通过一个贴图的例子，学习如何绘制位图图片和控制图片的旋转。


Note

（1）新建项目 EX12_2，其目录结构如图 12-3 所示。

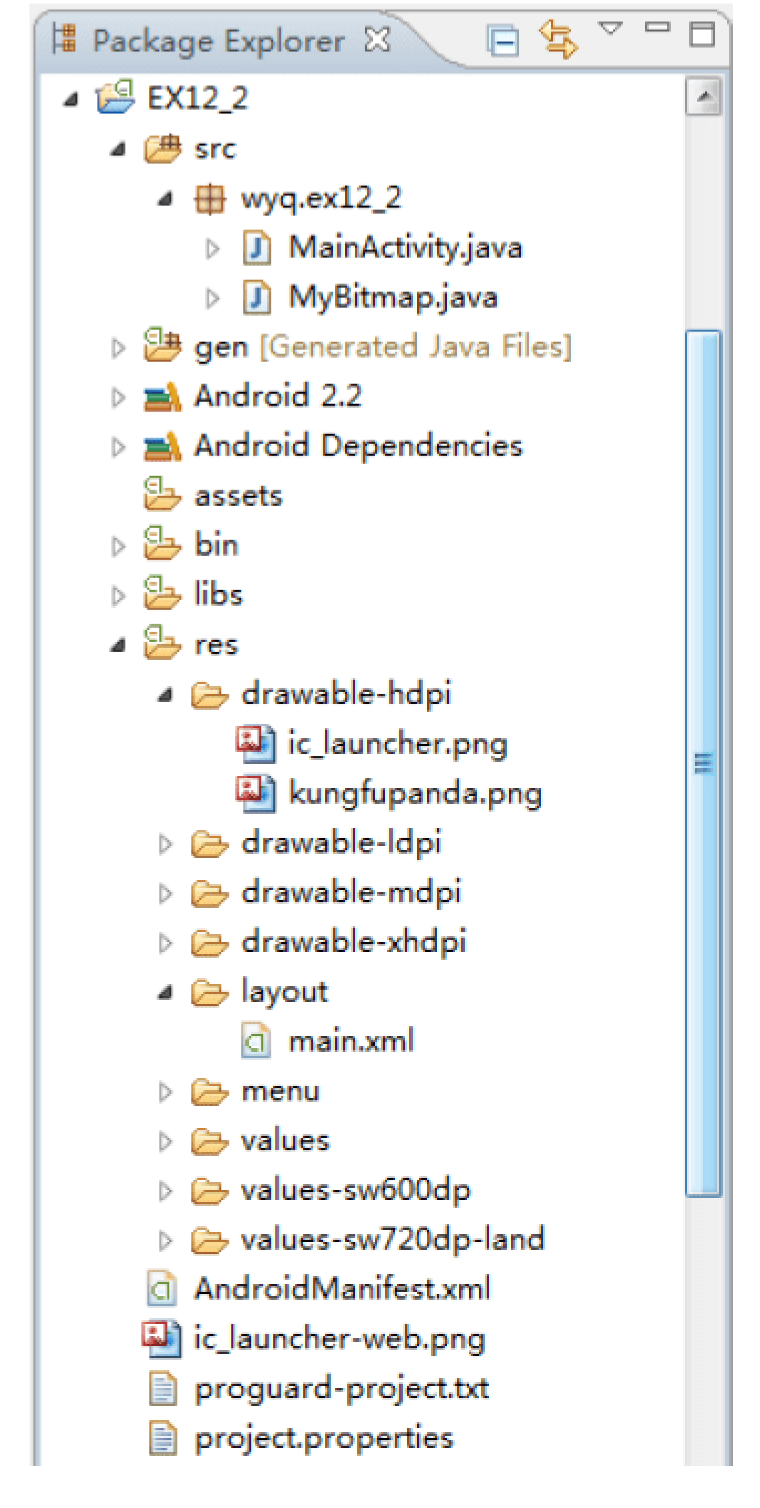


图 12-3　EX12_2 目录结构

（2）在 src 下创建两个类，一个是继承了 View，用来操作图片的 MyBitmap 类；另外一个是用来调用创建图片的 MainActivity 类。

① MainActivity 类的源代码如下：

```
1   public class MainActivity extends Activity {
2
3           @Override
4       protected void onCreate(Bundle savedInstanceState) {
5               super.onCreate(savedInstanceState);
6               setContentView(new MyBitmap (this));//调用自定义的 MyBitmap
7           }
8       }
```

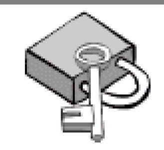
说明：

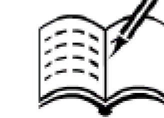
第 6 行： Activity 调用自定义的 MyBitmap 类。

游戏的核心是不断地绘图和刷新界面源，MainActivity 类主要用来调用我们自定义的视图。

② MyBitmap 类的源代码如下：

```
public class MyBitmap extends View {
    Bitmap myBitmap;                        //图片的引用
    Paint paint;                            //画刷的引用
```






```
4
5       public MyBitmap(Context context) {                          //构造器
6           super(context);
7           this.initBitmap();                                      //调用初始化方法
8       }
9
10      public void initBitmap() {
11          paint = new Paint();                                    //创建一个画刷
12          myBitmap = BitmapFactory.decodeResource(getResources(),
13                  R.drawable.kungfupanda);                        //获得图片资源
14      }
15
16      @Override
17      protected void onDraw(Canvas canvas) {                      //重写的绘制方法
18          super.onDraw(canvas);
19          paint.setAntiAlias(true);                               //设置抗锯齿效果
20          paint.setColor(Color.WHITE);                            //设置画刷的颜色
21          paint.setTextSize(15);                                  //设置字体大小
22          canvas.drawColor(Color.BLACK);                          //设置背景颜色
23          canvas.drawBitmap(myBitmap, 10, 10, paint);             //绘制图片
24          canvas.save();                                          //保存画布状态
25          Matrix m1 = new Matrix();                               //创建 Matrix 对象
26          m1.setTranslate(500, 10);                               //平移矩阵
27          Matrix m2 = new Matrix();                               //创建 Matrix 对象
28          m2.setRotate(15);                                       //旋转矩阵
29          Matrix m3 = new Matrix();                               //创建 Matrix 对象
30          m3.setConcat(m1, m2);                                   //设置两个 Matrix 的组合
31          m1.setScale(0.8f, 0.8f);                                //设置 Matrix 的缩放
32          m2.setConcat(m3, m1);                                   //设置两个 Matrix 的组合
33          canvas.drawBitmap(myBitmap, m2, paint);                 //绘制图片
34          canvas.restore();                                       //恢复画布状态
35          canvas.save();                                          //保存画布状态
36          paint.setAlpha(180);                                    //设置透明度
37          m1.setTranslate(200, 100);                              //平移矩阵
38          m2.setScale(1.3f, 1.3f);                                //设置 Matrix 的缩放
39          m3.setConcat(m1, m2);                                   //设置两个 Matrix 的组合
40          canvas.drawBitmap(myBitmap, m3, paint);                 //绘制图片
41          paint.reset();                                          //恢复画刷设置
42          canvas.restore();                                       //恢复画布状态
43          paint.setTextSize(40);                                  //设置字体大小
44          paint.setColor(0xffFFFFFF);                             //设置画刷颜色
45          canvas.drawText("图片的宽度: " + myBitmap.getWidth(), 20, 380, paint);//绘制字符串
46          canvas.drawText("图片的高度: " + myBitmap.getHeight(), 20, 430, paint);//绘制字符串
47          paint.reset();                                          //恢复画刷设置
48      }
49  }
```

说明：

- ❑ 第 7 行：View 下的构造函数中调用初始化方法 initBitmap()。

- 第 12 行：使用 BitmapFactory 来获取资源中的位图 kungfupanda.png。
- 第 17 行：重写 View 绘制方法 onDraw(Canvas canvas)。onDraw()方法会传入一个 Canvas 对象，是用来绘制控件视觉界面的画布。在 onDraw()方法里，经常会看到调用 save()和 restore()方法，作用如下。
  - save()：用来保存 Canvas 的状态。保存之后，可以调用 Canvas 的平移、缩放、旋转、错切、裁剪等操作。
  - restore()：用来恢复 Canvas 之前保存的状态。防止保存后对 Canvas 执行的操作影响后续的绘制。

  save()和 restore()要配对使用（restore()可以比 save()少，但不能多），如果 restore()调用次数比 save()多，会引发 Error。save()和 restore()之间往往夹杂着对 Canvas 的特殊操作。
- 第 23 行：借助于 BitmapFactory 获取位图，通过 Canvas 类的 drawBitmap 显示位图。
- 第 25~32 行：位图的旋转也可以借助 Matrix 来实现，Android SDK 提供了 Matrix 类，可以通过各种接口来设置矩阵。本例用到的方法有 setRotate()（旋转）、setScale()（缩放）、setTranslate()（平移）和 setConcat()（连接）。

（3）将 res\layout 下的 main.xml 的代码修改如下：

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       >
7       <!--添加一个自定义的 View 到线性布局中 -->
8       < wyq.ex12_2.MyBitmap
9           android:id="@+id/ myBitmap"
10          android:layout_width="match_parent"
11          android:layout_height="match_parent" />
12  </LinearLayout>
```

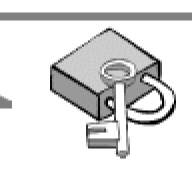
说明：

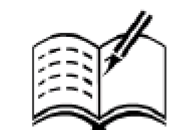
第 8~11 行：在垂直的线性布局中添加了一个自定义的 View，并指定了 ID，宽度和高度全部填充父控件。

项目 EX12_2 的运行结果如图 12-4 所示。



图 12-4　EX12_2 运行结果





12.2 贪吃蛇游戏的解析

贪吃蛇又叫贪食蛇，是一款非常经典的休闲单机小游戏，其玩法简单、耗时少，游戏的界面比较简单，逻辑实现也不复杂，适合 Android 初学者作为游戏入门的练习项目。用户使用方向键控制一条蛇来吃苹果，与此同时，蛇的身体也随着吃掉的苹果的数量增多而不断变长，蛇的行进速度随着游戏时间的增长而越来越快，当蛇头撞到自己的身体或四周的墙壁时游戏结束。

游戏实现流程如下：

（1）运行程序，首先进入的是 Android 系统虚拟手机的主界面。

（2）启动游戏后，进入欢迎界面，提示“press up to start……”，按下键盘 up 键开始游戏。

（3）游戏开始后，可以选择 Pause（暂停），也可以随时恢复游戏。

（4）游戏结束后，会自动显示本次游戏的得分情况，也可以重新开始游戏。

整个游戏流程图如图 12-5 所示。

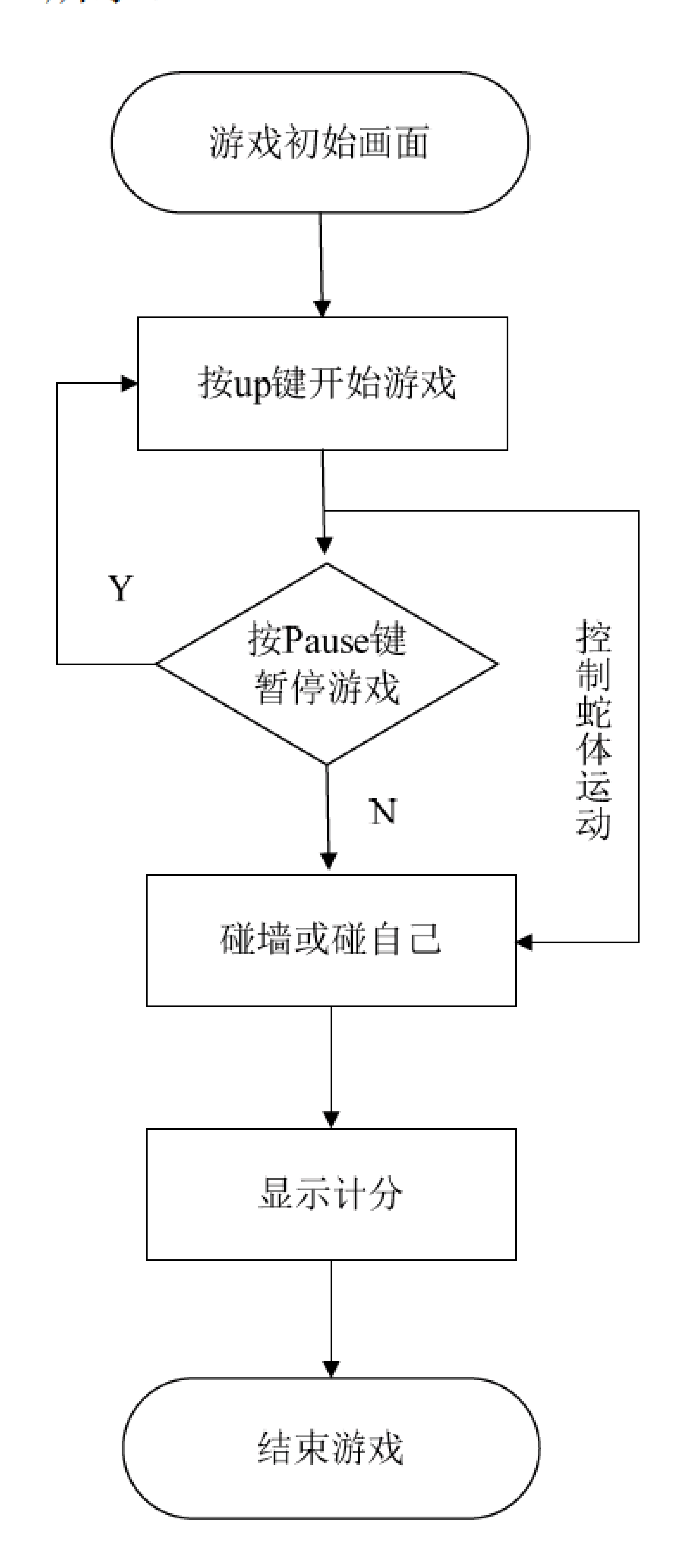
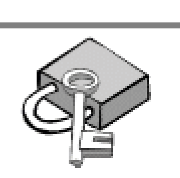
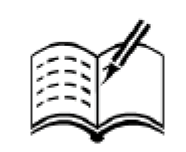


图 12-5　游戏流程图





12.2.1 在 Eclipse 中导入并运行游戏

打开 Eclipse，选择 File/New/Project 命令，单击项目 Android，选择 Android Sample Project，单击 Next 按钮，选中所用 SDK 版本，单击 Next 按钮，选择 Snake，创建贪吃蛇游戏应用项目，其结构如图 12-6 所示。

Snake 工程的源文件有 3 个：Snake.java、SnakeView.java 和 TileView.java。Snake 类是这个游戏的入口点，TitleView 类进行游戏的绘画，SnakeView 类则是对游戏控制操作的处理。Coordinate 和 RefreshHandler 是两个辅助类，也是 SnakeView 类的内部类。其中，Coordinate 是一个点的坐标(x, y)，RefreshHandler 将 RefreshHandler 对象绑定在某个线程并给它发送消息，如图 12-7 所示。

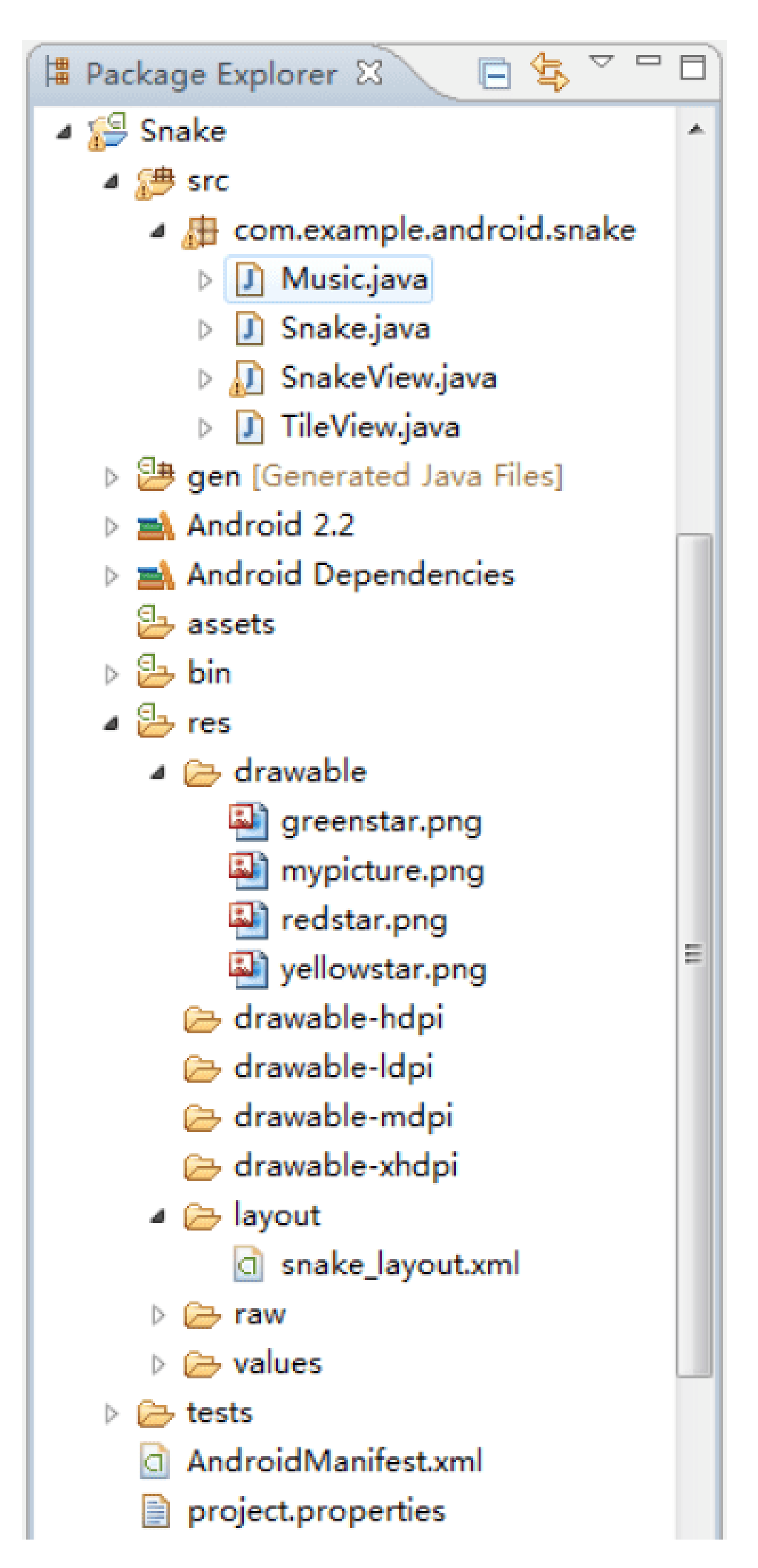


图 12-6 游戏应用项目结构

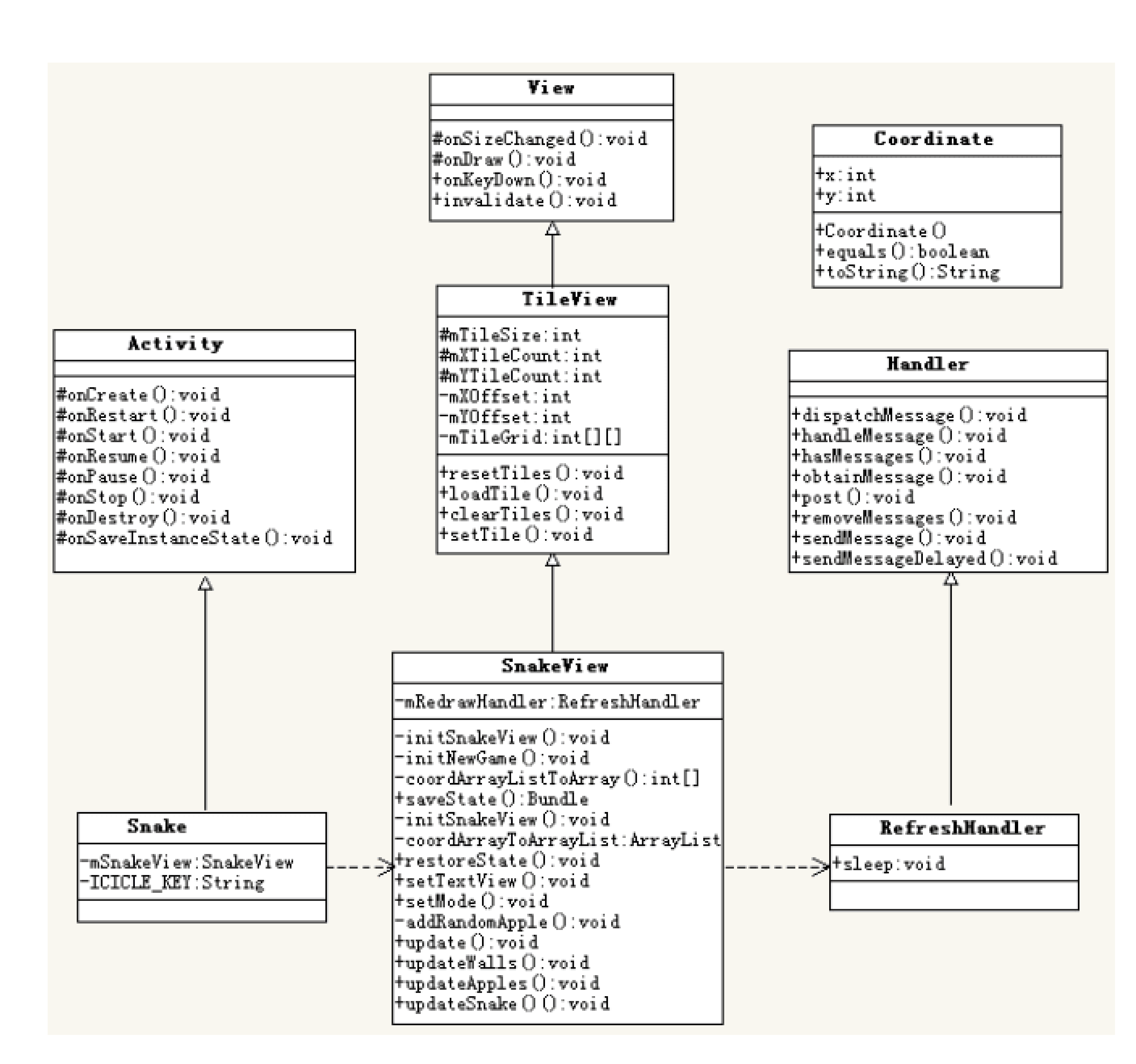
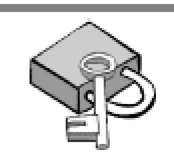


图 12-7 游戏的相关类图

运行游戏时，在虚拟设备上玩家不能用鼠标单击游戏画面进行控制，而只能通过按键来玩游戏。该游戏主要使用键盘上的方向键来进行控制，要想使计算机键盘的方向键操控虚拟设备有效，需要进行如下配置：

（1）打开 Eclipse，选择 Window/Android Virtual Device Manager 命令，或在工具栏单击 Android 虚拟设备管理的图标，打开 Android Virtual Device Manager 窗口，如图 12-8 所示。



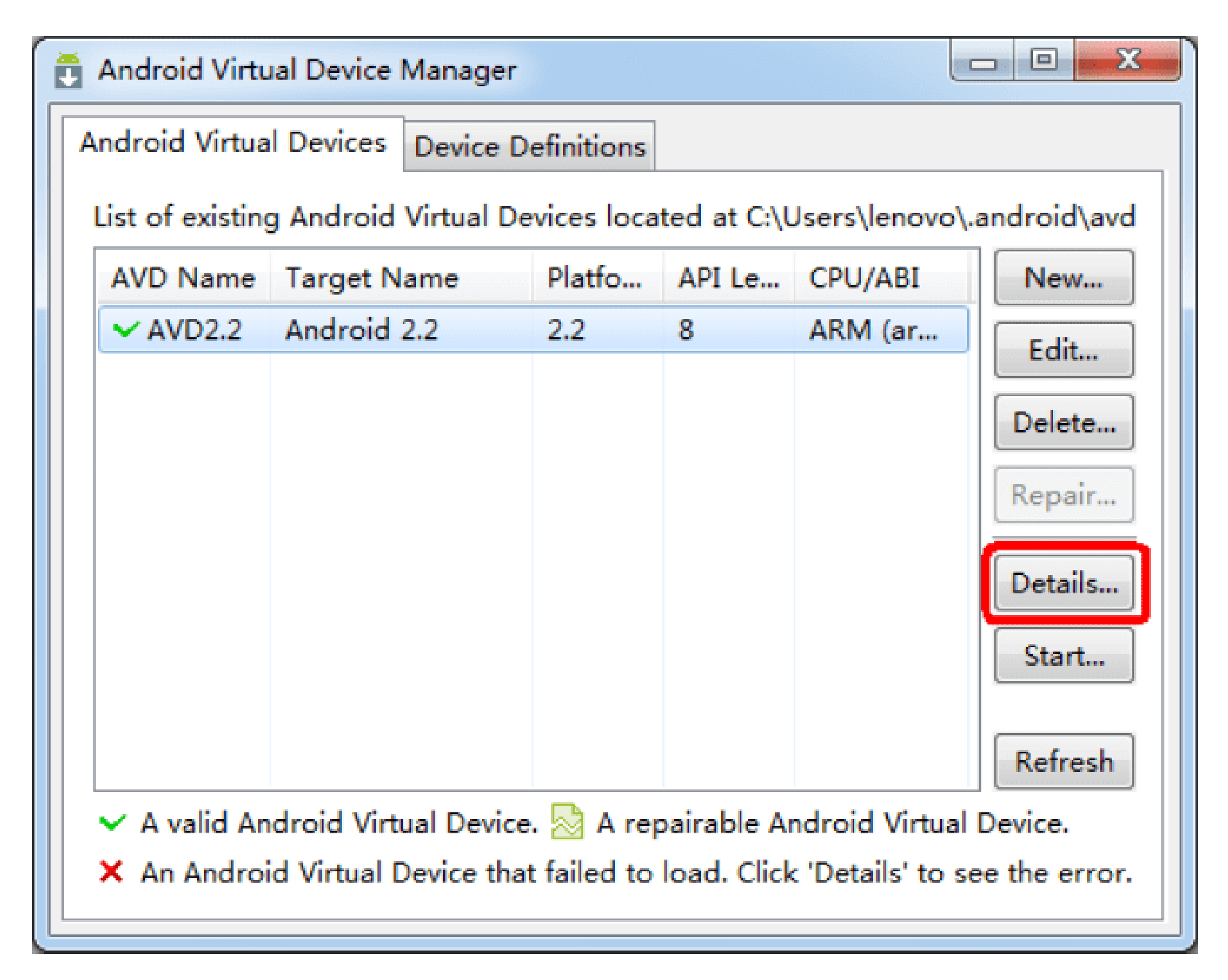


图 12-8　Android 虚拟设备管理的界面

（2）选中已经配置好的虚拟设备，单击 Details 按钮，弹出如图 12-9 所示的 Android 虚拟设备参数的配置细节。如果 hw.dPad 为 yes，则需要去在路径 C:\Users\lenovo\.android\avd\AVD2.2.avd（不同计算机配置路径可能不同）下找到 config.ini 文件，并把文件里的 hw.dPad=no 修改为 hw.dPad=yes，保存后重启 Eclipse。

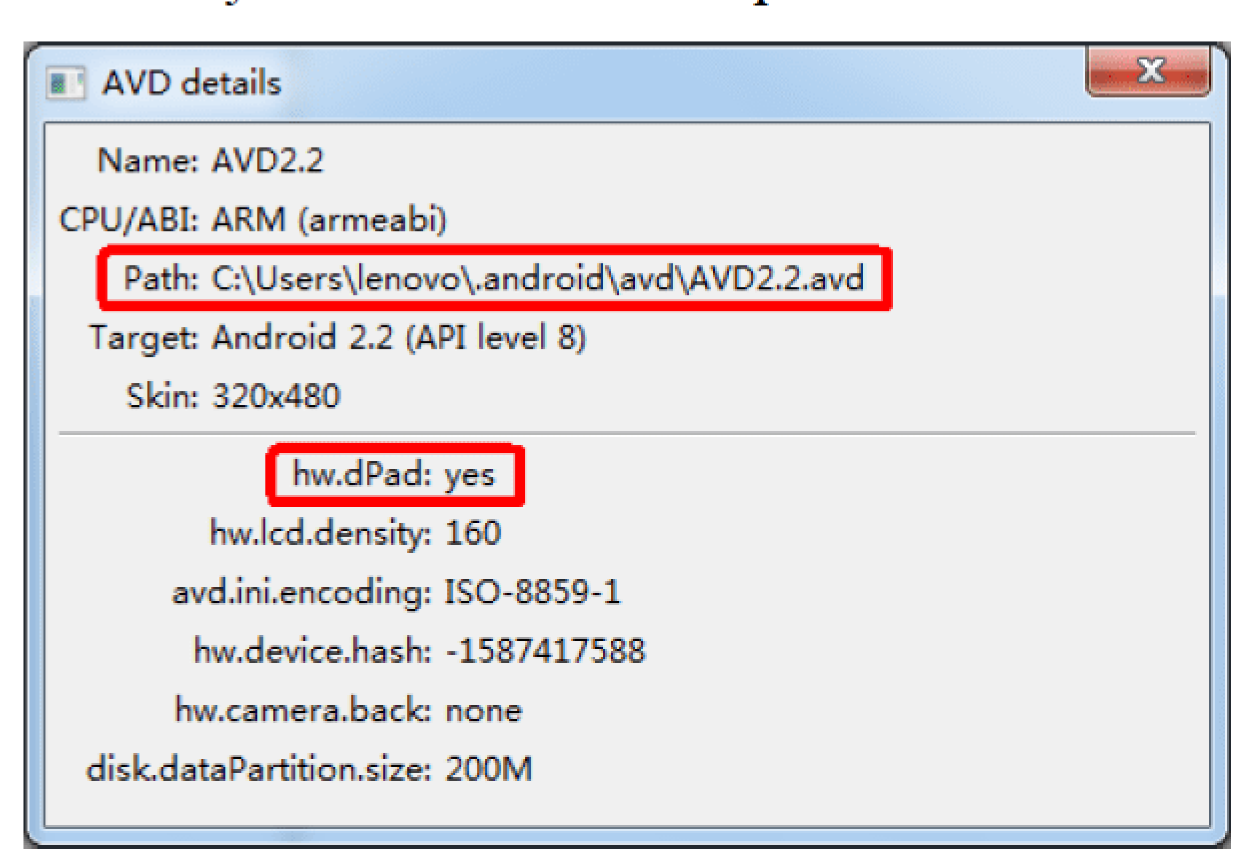
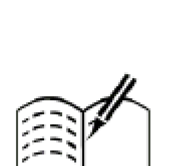


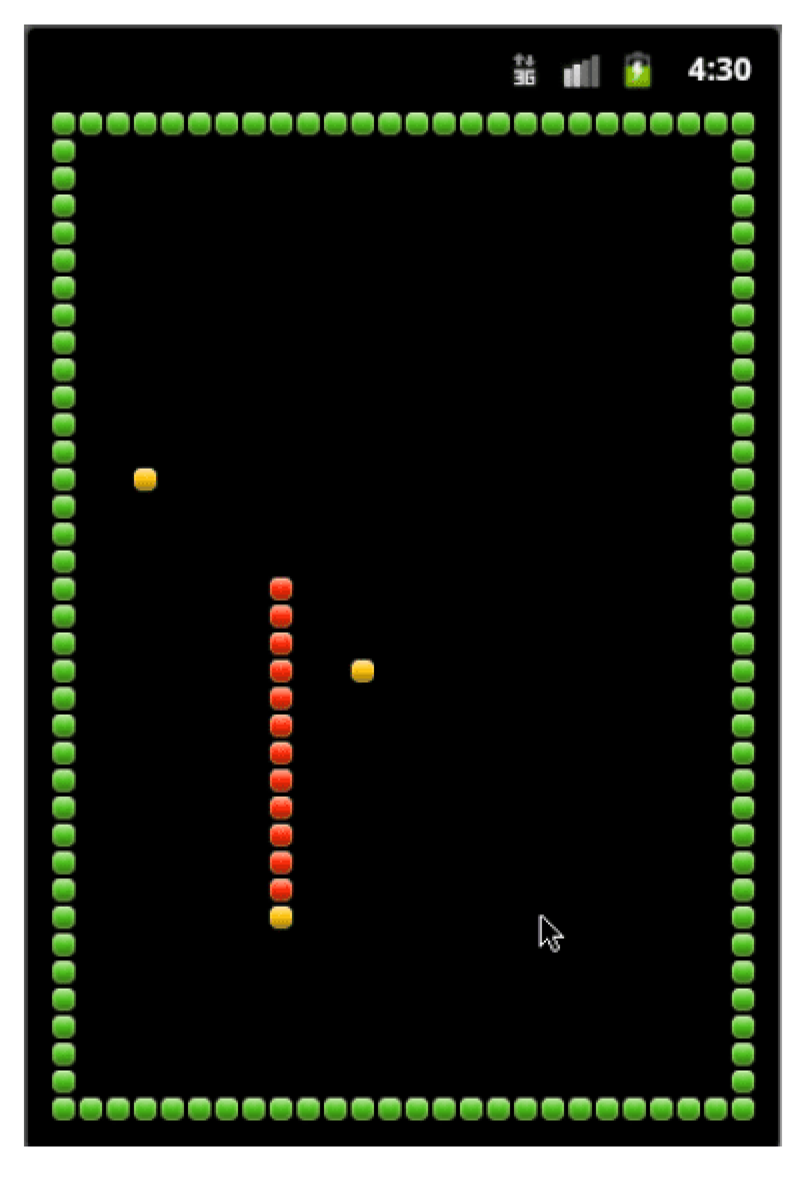
图 12-9　Android 虚拟设备参数配置

12.2.2　游戏界面布局

贪吃蛇游戏的玩法是，玩家只需要用上、下、左、右 4 个方向键操控一条贪吃蛇，使它不停地在屏幕上游走，吃各个方向出现的苹果，贪吃蛇的长度会随着所吃苹果数的增多而增长，速度也会加快。只要蛇头碰到屏幕四周或自己的身体，贪吃蛇就立即毙命，游戏结束，计算出分数。图 12-10 显示了贪吃蛇游戏的运行和结束界面。

下面介绍该游戏界面布局。





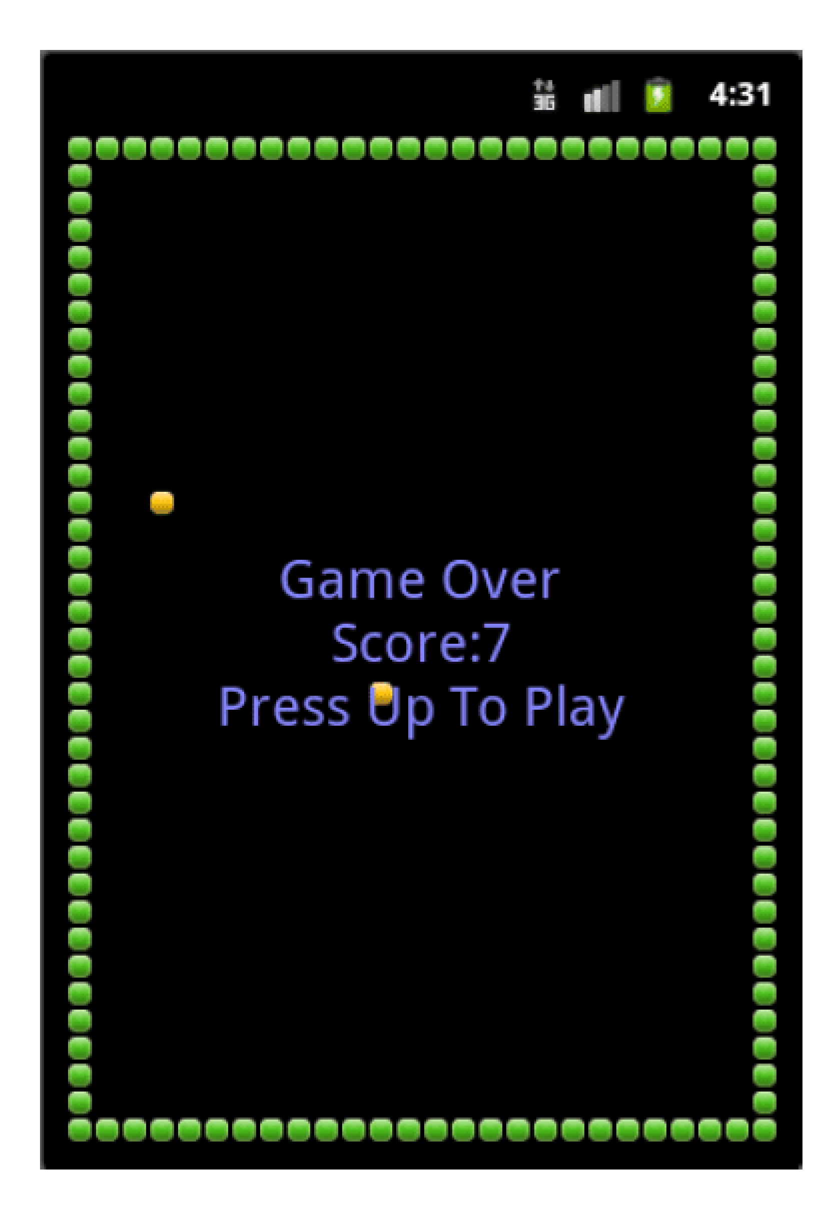
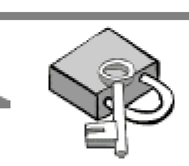
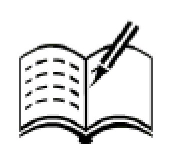


图 12-10　贪吃蛇游戏的运行和结束界面

（1）snake_layout.xml 布局文件的内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android=http://schemas.android.com/apk/res/android
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <!--添加一个自定义的 View 到帧布局中  -->
        <com.example.android.snake.SnakeView
          android:id="@+id/snake"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          tileSize="24"/>
        <!--添加一个相对布局到帧布局中  -->
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
            <!--添加一个 TextView 到相对布局中  -->
            <TextView
                android:id="@+id/text"
                android:text="@string/snake_layout_text_text"
                android:visibility="visible"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerInParent="true"
                android:gravity="center_horizontal"
                android:textColor="#ff8888ff"
                android:textSize="24sp"/>
        </RelativeLayout>
 </FrameLayout>
```





说明：

- 第 1 行：Snake 游戏项目使用一个 FrameLayout 布局，也叫帧布局或窗体布局。原理是在控件中先绘制的任何一个控件都可以被后绘制的控件覆盖，最后绘制的控件会覆盖之前的控件。
- 第 6~10 行：在 FrameLayout 布局下，先使用自定义资源标签方式，自定义的 SnakeView 必须写全名，即 com.example.android.snake，因为 R.资源不包含 SnakeView 类。设置与其他控件一样，指定了 ID，宽度和高度全部填充父控件和自定义的标签 tileSize（尾巴长度）。
- 第 12~26 行：使用 RelativeLayout（相对布局），设置一个 TextView。通过设置属性 android:layout_centerInParent="true"和 android:gravity="center_horizontal"，使文字相对于父元素完全居中和水平居中。

观察 Snake 的布局文件，可以理解为一个 SnakeView 和一个 TextView 启动后，后者会覆盖在前者上面。

游戏开始和暂停界面的效果如图 12-11 所示。

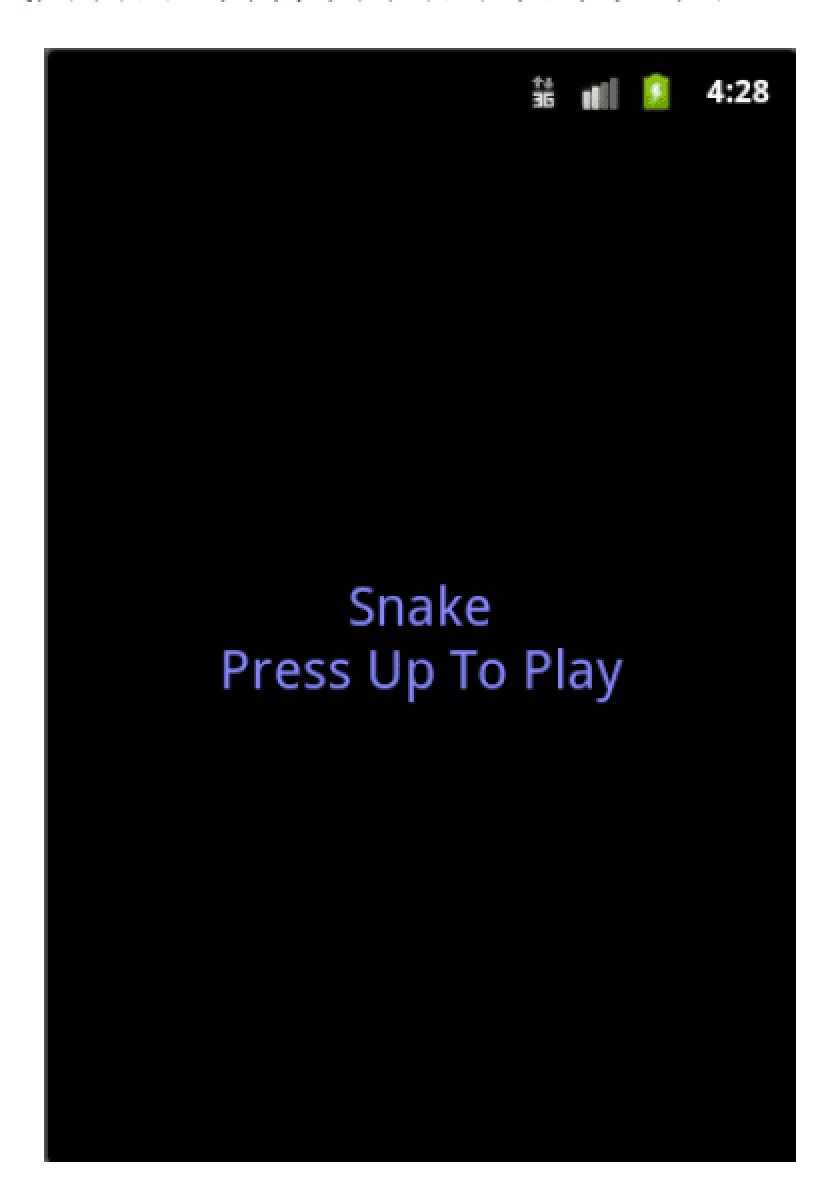




图 12-11　贪吃蛇游戏的开始和暂停界面

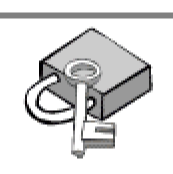
（2）对应布局文件代码的实现。

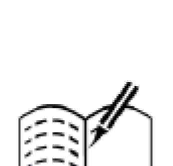
游戏的 Snake 为主 Activity 类，Snake.java 代码如下：

```
public class Snake extends Activity {
    private SnakeView mSnakeView;

    private static String ICICLE_KEY = "snake-view";

    @Override
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
```



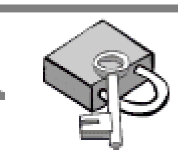


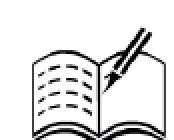

```
9
10              setContentView(R.layout.snake_layout);
11
12              mSnakeView = (SnakeView) findViewById(R.id.snake);
13              mSnakeView.setTextView((TextView) findViewById(R.id.text));
14
15              if (savedInstanceState == null) {
16                  mSnakeView.setMode(SnakeView.READY);
17              } else {
18                  Bundle map = savedInstanceState.getBundle(ICICLE_KEY);
19                  if (map != null) {
20                      mSnakeView.restoreState(map);
21                  } else {
22                      mSnakeView.setMode(SnakeView.PAUSE);
23                  }
24              }
25          }
26
27          @Override
28          protected void onPause() {
29              super.onPause();
30              mSnakeView.setMode(SnakeView.PAUSE);
31          }
32
33          @Override
34          public void onSaveInstanceState(Bundle outState) {
35              outState.putBundle(ICICLE_KEY, mSnakeView.saveState());
36          }
37  }
```

说明：

- 第 10~13 行：在 Snake 这个 Activity 的 onCreate()方法中，首先将 Layout 文件中的 SnakeView 和 TextView 关联起来。
- 第 15、16 行：设置 SnakeView 的状态为 Ready，存储状态为空，说明游戏刚启动，可以切换到准备状态。通过调用 SnakeView 类中的 setMode()函数设置游戏状态：

```
public void setMode(int newMode) {
    //把当前游戏状态存入 oldMode
    int oldMode = mMode;
    //把游戏状态设置为新状态
    mMode = newMode;
    //如果新状态是运行状态，且原有状态为不运行，那么就开始游戏
    if (newMode == RUNNING & oldMode != RUNNING) {
        mStatusText.setVisibility(View.INVISIBLE);
        update();
        return;
    }
```






```
12
13            Resources res = getContext().getResources();
14            CharSequence str = "";
15            //如果新状态是暂停状态，那么设置文本内容为暂停内容
16            if (newMode == PAUSE) {
17                str = res.getText(R.string.mode_pause);
18            }
19            //如果新状态是准备状态，那么设置文本内容为准备内容
20            if (newMode == READY) {
21                str = res.getText(R.string.mode_ready);
22            }
23           //如果新状态是失败状态，那么设置文本内容为失败内容
24            if (newMode == LOSE) {
25                str = res.getString(R.string.mode_lose_prefix) + mScore
26                      + res.getString(R.string.mode_lose_suffix);
27            }
28            //设置文本
29            mStatusText.setText(str);
30            //显示该 View
31            mStatusText.setVisibility(View.VISIBLE);
32        }
```

说明：

- 第 29 行：mStatusText 是 SnakeView 类定义的一个 TextView 私有变量，用来显示游戏状态的文本组件。
- 第 31 行：执行这一句后显示游戏的起始画面。

12.2.3 游戏界面实现部分

Snake 游戏项目把主界面分成界面 UI 和游戏逻辑两层，最基础的界面 UI 部分用父类 TileView 来表示，子类 SnakeView 是在继承 TileView 的 UI 基础上，加入相应的游戏控制逻辑，从而实现了两者的分离，这对于游戏的修改和扩展非常便利。

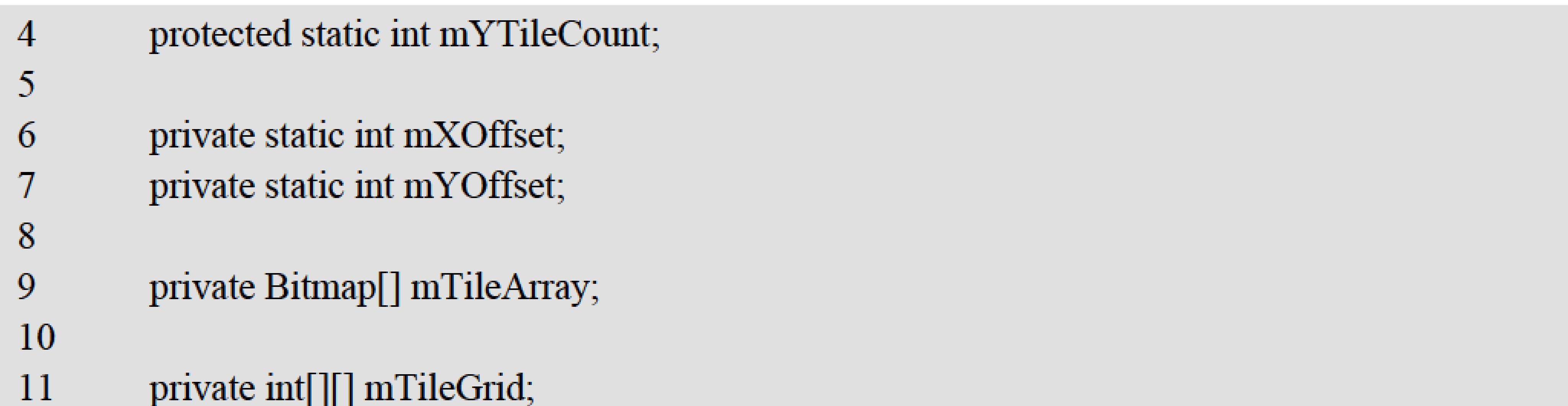
TileView 类，从名称上不难看出这是一个方块（格）图类，就是生成一个方块（格）。TileView 使用了 Android 平台的显示基类 View，View 类是直接从 java.lang.Object 派生出来的，是各种控件，如 TextView、EditView 的基类，当然包括窗口 Activity 类。

TileView 的界面 UI 部分，基本思想是把整个屏幕看作一个二维数组，每一个元素可以视为一个方块（格），因此每个方块（格）在游戏进行过程中可以处于不同的状态，如空闲、墙、苹果、贪吃蛇（蛇身或蛇头）。操作游戏的过程其实就是不断修改相应方格的状态，然后再让整个 View 重新绘制自身。当然，还需要加入一些游戏当前所处状态（失败或成功）的判定机制。

在 TileView.java 文件中，TileView 类的数据成员如下：

```
protected static int mTileSize;

protected static int mXTileCount;
```



```
4       protected static int mYTileCount;
5
6       private static int mXOffset;
7       private static int mYOffset;
8
9       private Bitmap[] mTileArray;
10
11      private int[][] mTileGrid;
```

说明：

- 第 1 行：定义方块（格）的数量。
- 第 3、4 行：定义 X 轴和 Y 轴方块（格）的数量。
- 第 6、7 行：定义 X 坐标和 Y 坐标的偏移量。
- 第 9 行：定义存储 3 种方块（格）的图标文件。
- 第 11 行：定义保存每个方块（格）的索引——二维数组。

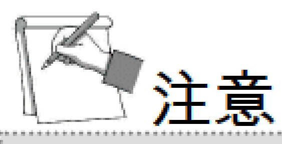
在游戏还未正式开始前，首先要做一些初始化工作，在 View 第一次加载时会首先调用 onSizeChanged()方法。代码如下：

```
1   protected void onSizeChanged(int w, int h, int oldw, int oldh) {
2           mXTileCount = (int) Math.floor(w / mTileSize);
3           mYTileCount = (int) Math.floor(h / mTileSize);
4
5           mXOffset = ((w - (mTileSize * mXTileCount)) / 2);
6           mYOffset = ((h - (mTileSize * mYTileCount)) / 2);
7
8           mTileGrid = new int[mXTileCount][mYTileCount];
9           clearTiles();
10      }
```

说明：

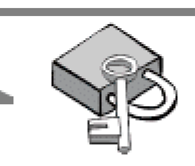
onSizeChanged()方法当 View 的尺寸改变时调用，是在 onDraw()方法调用之前就会被调用，用来设置一些变量的初始值，在视图大小改变时调用，如手机由垂直旋转为水平。

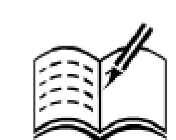
- 第 2、3 行：定义屏幕中可放置的方块（格）的行数和列数。
- 第 5、6 行：定义 X 轴和 Y 轴偏移量。
- 第 8 行：定义方块（格）的二维数组。
- 第 9 行：清空所有方格（块）。



注意

模拟器屏幕默认的像素是 320×480，而代码中默认的方块（格）大小为 12，因此屏幕上放置的方块（格）数为 26×40，把屏幕剖分后，再设置一个相应的二维整型数组来记录每一个方块（格）的状态，根据方块（格）的状态，可以从 mTileArray 保存的图标文件中读取对应的状态图标。同时也适应各种分辨率的屏幕，当改变屏幕大小尺寸时，同时修改





第一次调用完 onSizeChanged()后，会紧跟着第一次调用 onDraw()来绘制 View 自身，当然，此时由于所有方块（格）的状态都是 0，所以什么也不会绘制。onDraw()在视图需要重绘的时候调用，如使用 invalidate 刷新界面上的某个矩形区域。

在 TileView.java 文件中，onDraw()方法如下：

```
1   public void onDraw(Canvas canvas) {
2           super.onDraw(canvas);
3
4           for (int x = 0; x < mXTileCount; x += 1) {
5               for (int y = 0; y < mYTileCount; y += 1) {
6                   if (mTileGrid[x][y] > 0) {
7                       canvas.drawBitmap(mTileArray[mTileGrid[x][y]],
8                               mXOffset + x * mTileSize,
9                               mYOffset + y * mTileSize,
10                              mPaint);
11                  }
12              }
13          }
14  }
```

说明：

onDraw()要做的工作非常简单，就是扫描每一个方块（格），根据方块（格）当前状态，从图标文件中选择对应的图标绘制到这个方块（格）上。当然，onDraw()在游戏进行过程中会不断地被调用，从而界面不断被更新。

- 第 1 行：定义 onDraw()，在视图需要重绘时调用，如使用 invalidate 刷新界面上的某个矩形区域。
- 第 4、5 行：判断方块（格）位置状态索引大于 0，也就是不为空时。
- 第 6~10 行：mTileGrid 中不为 0 时画此方块（格）。

12.2.4 游戏逻辑部分

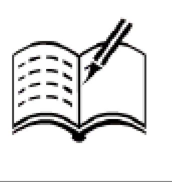
下面介绍子类 SnakeView 是如何在父类 TileView 的基础上加入特定的游戏逻辑，从而完成 Snake 程序的。

（1）初始化地图坐标。由于 SnakeView 类从 TileView 类继承而来，所以已经拥有二维方块（格）地图（只是此时地图里的所有方格状态都是 0）。要想初始化这个地图，可在 initNewGame 函数中实现。代码如下：

```
private ArrayList<Coordinate> mSnakeTrail = new ArrayList<Coordinate>();
private ArrayList<Coordinate> mAppleList = new ArrayList<Coordinate>();

private void initNewGame() {
        mSnakeTrail.clear();
        mAppleList.clear();

        mSnakeTrail.add(new Coordinate(7, 7));
```


Note

```
9            mSnakeTrail.add(new Coordinate(6, 7));
10           mSnakeTrail.add(new Coordinate(5, 7));
11           mSnakeTrail.add(new Coordinate(4, 7));
12           mSnakeTrail.add(new Coordinate(3, 7));
13           mSnakeTrail.add(new Coordinate(2, 7));
14
15           mNextDirection = NORTH;
16
17           addRandomApple();
18           addRandomApple();
19
20           mMoveDelay = 600;
21           mScore = 0;
22       }
```

说明：

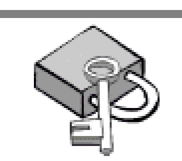
- 第 1 行：定义蛇身数组（数组以坐标对象为元素）。
- 第 2 行：定义苹果数组（数组以坐标对象为元素）。
- 第 5、6 行：清空蛇和苹果占据的方块（格）。
- 第 8~13 行：创建蛇身，组成固定的蛇的方块（格）。
- 第 15 行：方向也固定朝北。
- 第 17、18 行：两个随机位置的苹果。
- 第 20 行：设置移动延迟。对于系统来说，它可不断调用 onDraw()，onDraw()负责对整个屏幕进行绘制，那么系统是隔多长时间去调用 onDraw()函数？于是成员变量 mMoveDelay 此时就发挥作用了，通过它可以设置休眠的时间，时间一到，马上就会通知 SnakeView 去重绘制。
- 第 21 行：设置初始得分（mscore）为 0。

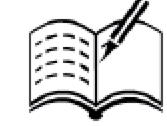
（2）加载图片，将用不同类型的图片绘制墙、贪吃蛇和苹果。代码如下：

```
1    private void initSnakeView() {
2            setFocusable(true);
3            Resources r = this.getContext().getResources();
4
5            resetTiles(4);
6
7            loadTile(RED_STAR, r.getDrawable(R.drawable.redstar));
8            loadTile(YELLOW_STAR, r.getDrawable(R.drawable.yellowstar));
9            loadTile(GREEN_STAR, r.getDrawable(R.drawable.greenstar));
10   }
```

说明：

- 第 1 行：定义 initSnakeView()函数。
- 第 2 行：定义可选焦点。
- 第 5 行：设置贴片图片数组。
- 第 7~9 行：把 3 种图片存到 Bitmap 对象数组。





Note

（3）调用为地图数组赋值的方法。代码如下：

```
//给某个方块（格）位置设置一个状态索引
    public void setTile(int tileindex, int x, int y) {
            mTileGrid[x][y] = tileindex;
    }
```

（4）初始化边界墙和苹果。计算出边界，对 map 进行赋值。代码如下：

```
1   private void updateWalls() {
2               for (int x = 0; x < mXTileCount; x++) {
3                    setTile(GREEN_STAR, x, 0);
4                    setTile(GREEN_STAR, x, mYTileCount - 1);
5               }
6               for (int y = 1; y < mYTileCount - 1; y++) {
7                    setTile(GREEN_STAR, 0, y);
8                    setTile(GREEN_STAR, mXTileCount - 1, y);
9               }
10        }
11
12        private void updateApples() {
13              for (Coordinate c : mAppleList) {
14                    setTile(YELLOW_STAR, c.x, c.y);
15              }
16    }
```

说明：

- 第 1 行：定义更新墙函数 updateWalls()。
- 第 3 行：给上边线的每个方块（格）位置设置一个绿色索引标识。
- 第 4 行：给下边线的每个方块（格）位置设置一个绿色索引标识。
- 第 7 行：给左边线的每个方块（格）位置设置一个绿色索引标识。
- 第 8 行：给右边线的每个方块（格）位置设置一个绿色索引标识。
- 第 12 行：更新苹果函数 updateApples()。

（5）绘制墙（边界）由 TitleView 类的 onDraw()方法实现。实际上，蛇的走动和苹果的出现就是由此方法绘制出来。关键是给地图数组赋值，赋值不同，边界、蛇、苹果不断地重绘，就动起来了。当蛇吃掉一个苹果后，蛇的长度增加，保存蛇的坐标信息的是 ArrayList。在线程中调用 Handler()方法，实现不断重绘。

（6）产生随机苹果，需要进行冲突检查。代码如下：

```
    private void addRandomApple() {
            Coordinate newCoord = null;
            boolean found = false;

              while (!found) {
                    int newX = 1 + RNG.nextInt(mXTileCount - 2);
                    int newY = 1 + RNG.nextInt(mYTileCount - 2);

```

Note

```
9                    newCoord = new Coordinate(newX, newY);
10
11                   boolean collision = false;
12                   int snakelength = mSnakeTrail.size();
13                   for (int index = 0; index < snakelength; index++) {
14                            if (mSnakeTrail.get(index).equals(newCoord)) {
15                           collision = true;
16                       }
17                   }
18                   found = !collision;
19               }
20               if (newCoord == null) {
21                   Log.e(TAG, "Somehow ended up with a null newCoord!");
22               }
23
24               mAppleList.add(newCoord);
25           }
```

说明：

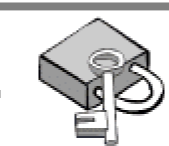
- ❑ 第 1 行：定义添加苹果函数 addRandomApple()。
- ❑ 第 2 行：定义新的坐标。
- ❑ 第 3 行：防止新苹果出现在蛇身下。
- ❑ 第 5 行：没有找到合适的苹果，就在循环体内一直循环，直到找到合适的苹果。
- ❑ 第 6、7 行：为苹果再找一个坐标，随机产生一个 X 值和 Y 值。
- ❑ 第 9 行：设置新坐标。
- ❑ 第 11 行：确保新苹果不在蛇身下，先假设没有发生冲突。
- ❑ 第 14 行：只要和蛇占据的任何一个坐标相同，即认为发生了冲突。
- ❑ 第 18 行：如果有冲突就继续循环，没有冲突则 found 的值为 true，退出循环，产生新坐标。
- ❑ 第 24 行：生成一个新苹果放在苹果列表中。

（7）蛇吃了苹果，长度增加且速度变快。代码如下：

```
int applecount = mAppleList.size();
for (int appleindex = 0; appleindex < applecount; appleindex++) {
    Coordinate c = mAppleList.get(appleindex);
    if (c.equals(newHead)) {
        mAppleList.remove(c);
        addRandomApple();
        mScore++;
        mMoveDelay *= 0.9;
        growSnake = true;
    }
}

mSnakeTrail.add(0, newHead);

```




```
15          if (!growSnake) {
16                  mSnakeTrail.remove(mSnakeTrail.size() - 1);
17          }
18          int index = 0;
19
20          for (Coordinate c : mSnakeTrail) {
21              if (index == 0) {
22                  setTile(YELLOW_STAR, c.x, c.y);
23              } else {
24                  setTile(RED_STAR, c.x, c.y);
25              }
26              index++;
27          }
28      }
```

说明：

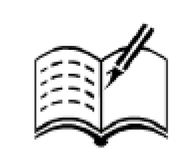
- 第 4 行：判断新蛇头是否与苹果重叠。
- 第 5 行：如果重叠，苹果坐标从苹果列表中移除。
- 第 6 行：立刻增加一个新苹果。
- 第 7 行：得分加 1。
- 第 8 行：延迟是以前的 90%，蛇提速。
- 第 9 行：蛇增长标志改为真。
- 第 13 行：在蛇头的位置增加一个新坐标。
- 第 15、16 行：如果蛇没增长，则删去最后一个坐标，相当于蛇向前走了一步。
- 第 20~27 行：重新设置颜色，蛇头是黄色的（同苹果一样），蛇身是红色的。

（8）蛇碰到墙或自身，游戏结束。代码如下：

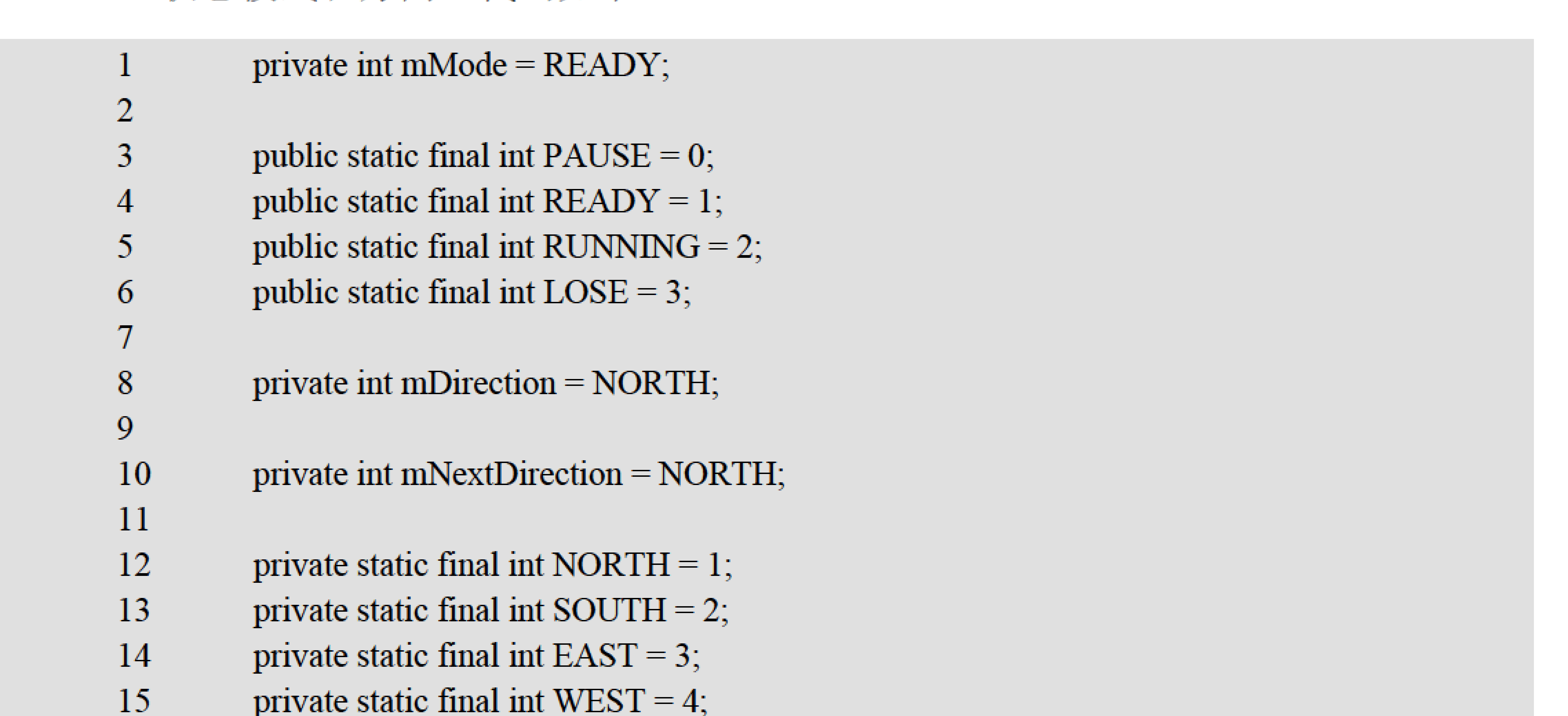
```
1         if ((newHead.x < 1) || (newHead.y < 1) || (newHead.x > mXTileCount - 2)
2                   || (newHead.y > mYTileCount - 2)) {
3                   setMode(LOSE);
4               return;
5           }
6
7           int snakelength = mSnakeTrail.size();
8           for (int snakeindex = 0; snakeindex < snakelength; snakeindex++) {
9               Coordinate c = mSnakeTrail.get(snakeindex);
10              if (c.equals(newHead)) {
11                      setMode(LOSE);
12                  return;
13              }
14         }
```

说明：

- 第 1 行：冲突检测。如果新蛇头与四面墙有重叠，那么游戏结束。
- 第 3 行：设置游戏状态为 LOSE，返回。
- 第 10 行：冲突检测。如果新蛇头与自身坐标重叠，游戏结束。



（9）状态模式和方向。代码如下：



```
1    private int mMode = READY;
2
3    public static final int PAUSE = 0;
4    public static final int READY = 1;
5    public static final int RUNNING = 2;
6    public static final int LOSE = 3;
7
8    private int mDirection = NORTH;
9
10   private int mNextDirection = NORTH;
11
12   private static final int NORTH = 1;
13   private static final int SOUTH = 2;
14   private static final int EAST = 3;
15   private static final int WEST = 4;
```

说明：

整个游戏分 READY、PAUSE 、RUNNING 、LOSE 4 种状态模式（mMode），分别对应准备、暂停、运行、结束。

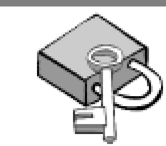
蛇的运行分 NORTH、SOUTH 、EAST、WEST 4 个方向，分别对应北、南、东、西 4 个方向，其中变量 mDirection 对应当前前进的方向，而 mNextDirection 对应下一步的移动方向。

- 第 1 行：游戏状态，默认值是准备状态。
- 第 3~6 行：游戏的 4 个状态，分别为暂停、准备、运行和结束。
- 第 8 行：游戏中蛇的前进方向，默认值为北方。
- 第 10 行：下一步的移动方向，默认值为北方。
- 第 12~15 行：游戏方向设定，包括北、南、东和西。

（10）按键处理。按键控制重构了 onKeyDown()方法，并且根据游戏不同的阶段对按键有不同的响应，游戏有 READY、PAUSE、RUNNING、LOSE 4 种状态模式。游戏用另外一个变量保存当前 snake 进行的方向，当游戏行进中时，按哪个方向就保存哪个方向。但是游戏规则同时规定，上下方向与左右方向不能相互交换。代码如下：

```
public boolean onKeyDown(int keyCode, KeyEvent msg) {
    if (keyCode == KeyEvent.KEYCODE_DPAD_UP) {
            if (mMode == READY | mMode == LOSE) {
             initNewGame();
             setMode(RUNNING);
             update();
             return (true);
          }

        if (mMode == PAUSE) {
             setMode(RUNNING);
```






```
12                  update();
13                  return (true);
14              }
15
16              if (mDirection != SOUTH) {
17                  mNextDirection = NORTH;
18              }
19              return (true);
20          }
21
22          if (keyCode == KeyEvent.KEYCODE_DPAD_DOWN) {
23              if (mDirection != NORTH) {
24                  mNextDirection = SOUTH;
25              }
26              return (true);
27          }
28
29          if (keyCode == KeyEvent.KEYCODE_DPAD_LEFT) {
30              if (mDirection != EAST) {
31                  mNextDirection = WEST;
32              }
33              return (true);
34          }
35
36          if (keyCode == KeyEvent.KEYCODE_DPAD_RIGHT) {
37              if (mDirection != WEST) {
38                  mNextDirection = EAST;
39              }
40              return (true);
41          }
42
43          return super.onKeyDown(keyCode, msg);
44      }
```

说明：

- 第 1 行：onKeyDown()方法，监听用户键盘操作，并处理这些操作。按键事件处理，确保贪吃蛇只能 90° 转向，而不能 180° 转向。
- 第 2 行：按下向上键。
- 第 3 行：准备状态或者失败状态时。
- 第 4 行：初始化游戏。
- 第 5 行：设置游戏状态为运行。
- 第 6 行：更新。
- 第 7 行：返回。
- 第 10 行：暂停状态时。
- 第 11 行：设置成运行状态。
- 第 16 行：运行状态时，如果原有方向不是向下，那么方向转向上。

Note

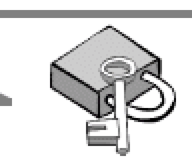
- 第 22 行：按下向下键。
- 第 23 行：原方向不是向上时，方向转向下。
- 第 29 行：按下向左键。
- 第 30 行：原方向不是向右时，方向转向左。
- 第 36 行：按下向右键。
- 第 37 行：原方向不是向左时，方向转向右。
- 第 43 行：按其他键时按原有功能返回。

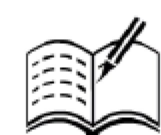
（11）游戏数据的保存机制。考虑到 Activity 的生命周期，如果用户在游戏期间离开游戏界面，游戏暂停；或者由于内存比较紧张，Android 关闭游戏释放内存，那么当用户返回游戏界面时恢复到上次离开时的界面。代码如下：

```
1       public void onCreate(Bundle savedInstanceState) {
2           super.onCreate(savedInstanceState);
3           setContentView(R.layout.snake_layout);
4           mSnakeView = (SnakeView) findViewById(R.id.snake);
5           mSnakeView.setTextView((TextView) findViewById(R.id.text));
6
7           if (savedInstanceState == null) {
8               mSnakeView.setMode(SnakeView.READY);
9           } else {
10                  Bundle map = savedInstanceState.getBundle(ICICLE_KEY);
11              if (map != null) {
12                  mSnakeView.restoreState(map);
13              } else {
14                  mSnakeView.setMode(SnakeView.PAUSE);
15              }
16          }
17      }
18
19      protected void onPause() {
20          super.onPause();
21          mSnakeView.setMode(SnakeView.PAUSE);
22      }
23
24      public void onSaveInstanceState(Bundle outState) {
25              outState.putBundle(ICICLE_KEY, mSnakeView.saveState());
26      }
27  }
```

说明：

- 第 1 行：onCreate()方法只会在 Activity 第一次创建时被调用。
- 第 7 行：检查存储状态以确定是重新开始还是恢复状态。
- 第 8 行：存储状态为空，说明游戏刚启动可以切换到准备状态。
- 第 10 行：已经保存过，那么恢复原有状态。
- 第 12 行：恢复状态。





- 第 14 行：设置状态为暂停。
- 第 19 行：暂停事件被触发时。
- 第 24 行：状态保存。
- 第 25 行：存储游戏状态到 View 里。

保存机制顺序图如图 12-12 所示。

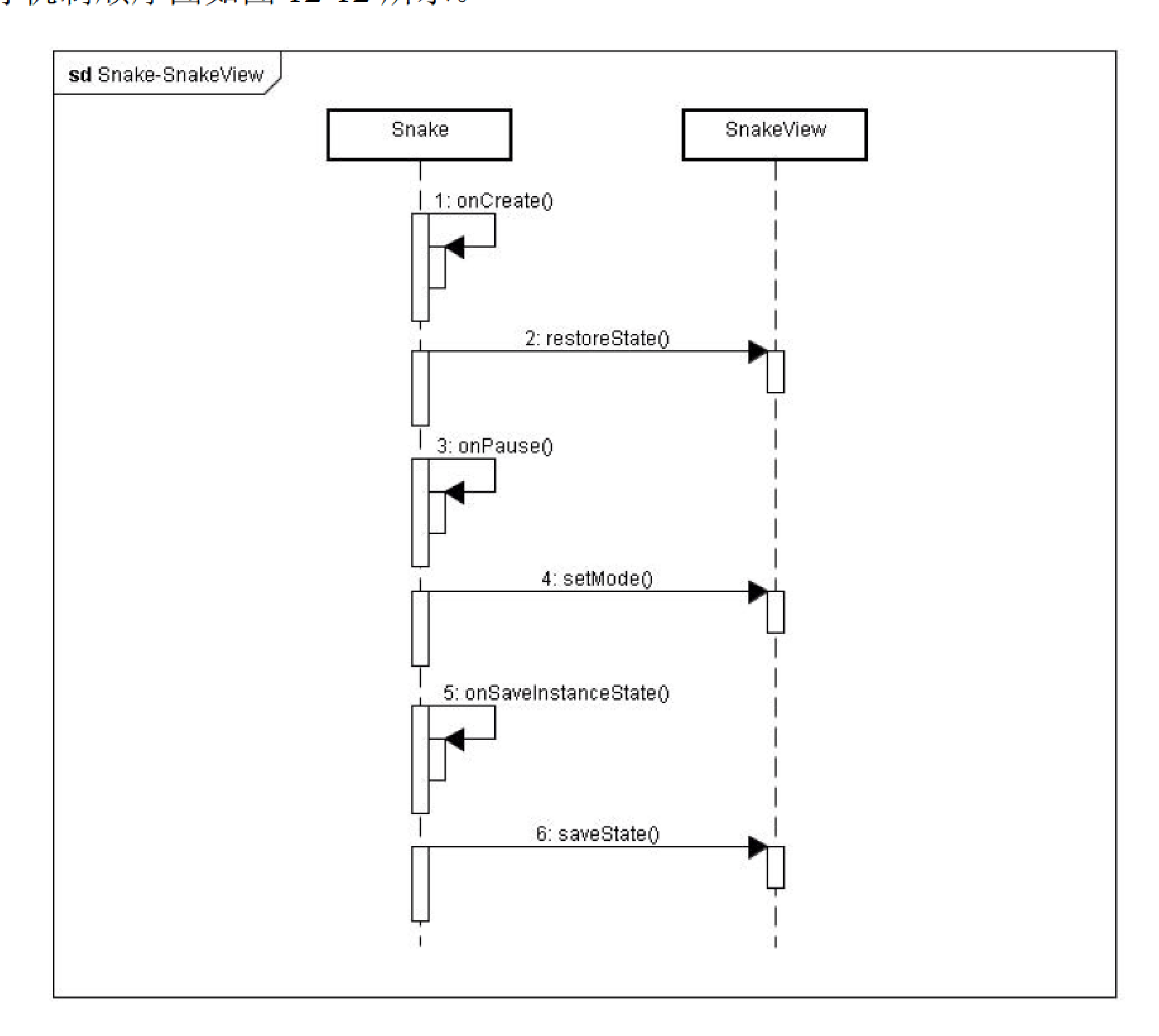


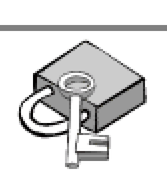
图 12-12　游戏数据的保存机制顺序图

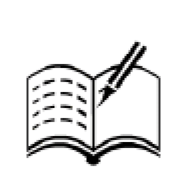
12.2.5　游戏引擎

任何游戏都需要有引擎来推动游戏的运行，最简单的游戏引擎是在一个线程中 while 循环，检测用户操作，对用户的操作做出反应，更新游戏的界面，直到用户退出游戏。

写过 JavaScript 或者 ActionScript 的开发者对于 setInterval()方法（按照指定的周期来调用函数或计算表达式）的用法会非常了解。那么在 Android 中如何实现 setInterval()方法呢？其中有两种方法可以实现类似的功能，一是在线程中调用 Handler()方法，二是应用 Timer，本例中使用了前者。

在 Snake 游戏中，辅助类 RefreshHandler 继承自 Handler，用来把 RefreshHandler 与当前线程进行绑定，从而可以直接给线程发送消息并处理消息。注意一点：Handler 对消息的处理都是异步的。RefreshHandler 在 Handler 的基础上增加 sleep()接口，用来每隔一个时间段给当前线程发送一个消息。handleMessage()方法在接收消息后，根据当前的游戏状态重绘界面，运行机制如图 12-13 所示。





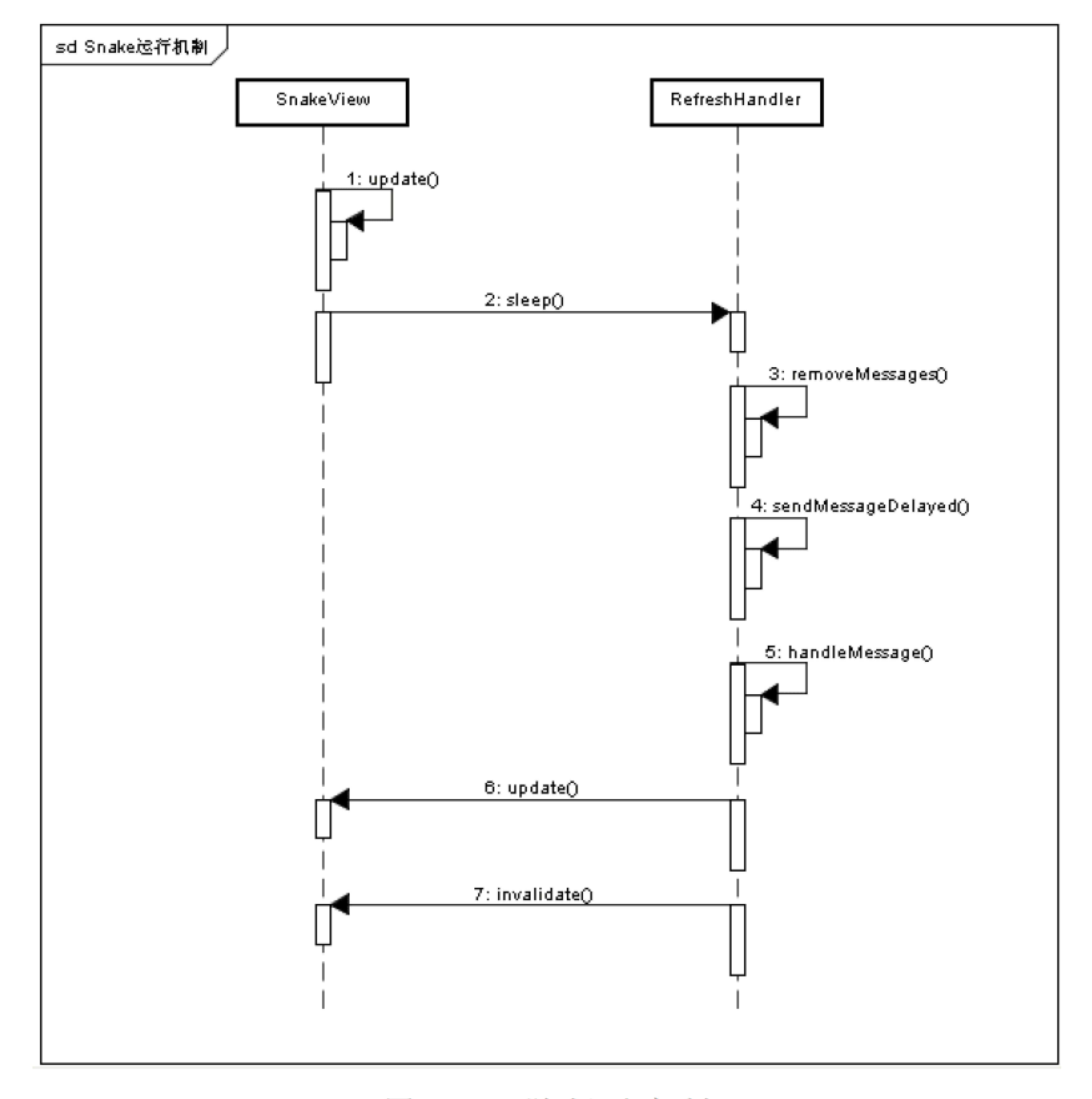


图 12-13　游戏运行机制

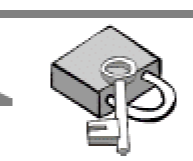
这类似于定时器的概念，在特定的时刻发送消息，根据消息处理相应的事件。update()与 sleep()间接的相互调用就构成了一个循环。这里要注意：mRedrawHandle 绑定的是 Activity 所在的线程，也就是程序的主线程；另外，由于 sleep()是个异步函数，所以 update()与 sleep()之间的相互调用才没有构成死循环。

Android 中实现 view 的更新有两组方法：invalidate()和 postInvalidate()，其中前者在 UI 线程自身中使用，而后者在非 UI 线程中使用。

invalidate()和 postInvalidate()方法需要使用 Android 提供的 handler，才能实现重绘，具体做法是在需要重绘的地方调用 handler 的 sendMessage()方法发送消息，紧接着 OS 会触发 handler 中的 handlerMessage()方法，在 handlerMessage()方法中再调用 view 的 invalidate()或者 postInvalidate()方法，并且由 invalidate()方法调用 onDraw()方法就能实现重绘。

```
class RefreshHandler extends Handler {
        public void handleMessage(Message msg) {
                SnakeView.this.update();
                SnakeView.this.invalidate();
        }

    public void sleep(long delayMillis) {
```



```
8                   this.removeMessages(0);
9                   sendMessageDelayed(obtainMessage(0), delayMillis);
10          }
11      };
```

说明：

- ❑ 第 1 行：定义一个 Handler。
- ❑ 第 2 行：处理消息队列。
- ❑ 第 3 行：更新 View 对象。
- ❑ 第 4 行：强制重绘。
- ❑ 第 7 行：延迟发送消息。休眠 delayMillis 毫秒。

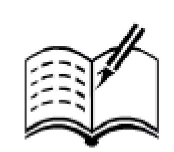
实际调用的处理函数 update()就可以说是整个游戏的引擎，正是由于它的工作（修改蛇和苹果的状态到一个新的状态，然后休眠自己，苏醒后在 Handler 中就会让系统区绘制上次修改过的二维方块地图，然后再次调用 update()，如此循环反复），才使得游戏不断被推进，因此，比作“引擎”很贴切。

```
1       public void update() {
2           if (mMode == RUNNING) {
3               long now = System.currentTimeMillis();
4
5               if (now - mLastMove > mMoveDelay) {
6                   clearTiles();
7                   updateWalls();
8                   updateSnake();
9                   updateApples();
10                  mLastMove = now;
11              }
12
13              mRedrawHandler.sleep(mMoveDelay);
14          }
15      }
```

说明：

- ❑ 第 1 行：更新各种动作，特别是贪吃蛇的位置，还包括墙、苹果等的更新。
- ❑ 第 2 行：如果是处于运行状态。
- ❑ 第 5 行：如果当前时间距离最后一次移动的时间超过了延迟时间。
- ❑ 第 13 行：Handler 会话进程 sleep 一个延迟时间单位。

既然 update()是游戏的动力，要让游戏停止下来，只要不再调用 update()即可，因为此时其实是画面静止了，游戏进入“暂停状态”。而这个状态还可以转为“运行状态”，即可以继续修改，再绘制游戏画面，如果进入“结束状态”，此时二维方块地图还停留在最后一个画面处，所以在游戏开始时要首先清理掉整个地图）。读者通过设置的断点即可观察到上次游戏运行时的底层数据。


Note

12.3 贪吃蛇游戏的功能拓展

Android SDK Sample 中的 Snake 工程虽然搭建了游戏的基本架构，但功能上仍不完善，用户体验尚显不足，读者可以在此基础上进行功能的拓展和改进，使游戏更加完美，进一步改善用户体验。

12.3.1 增加游戏的触摸控制

Android SDK Sample 中的 Snake 工程，在虚拟设备上玩家不能用鼠标单击游戏画面进行控制，而只能通过按键来玩游戏，在手机上不能使用触摸屏十分不方便，为此可以增加触摸控制功能。

实现鼠标单击和触摸功能并不复杂，只需要在 TileView.Java 文件中加入屏幕宽度和高度的信息声明，同时在 onSizeChanged()方法中获取屏幕宽度和高度的值即可。

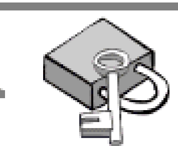
```
//屏幕宽度和高度信息
protected int width, height;
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        //获取屏幕宽度和高度的值
        width = w;
        height = h;
    …
}
```

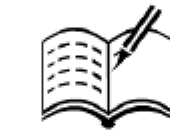
在 SnakeView.java 中，重写 View 中响应触摸事件的方法 onTouchEvent()，就可以实现鼠标单击和触摸功能。在重写的 onTouchEvent()方法中拦截用户触摸屏幕的一些信息，比如触摸屏幕的 X、Y 坐标，触摸屏幕发生的事件（包括触摸按下、触摸抬起和触摸移动），触摸屏幕发生的时间等。代码如下：

```
public boolean onTouchEvent(MotionEvent event)
  {
        if (mMode == READY | mMode == LOSE) {
              initNewGame();
              setMode(RUNNING);
              update();
              return (true);
        }

        int x = (int)event.getX();
        int y = (int)event.getY();

        if((mDirection) == 1||(mDirection) == 2) {
            if(x < width/2)
            {
                mNextDirection = WEST;
```






```
17                }
18                else
19                {
20                    mNextDirection = EAST;
21                }
22            }
23            else
24            {
25                if(y < height/2)
26                {
27                    mNextDirection = NORTH;
28                }
29                else
30                {
31                    mNextDirection = SOUTH;
32                }
33            }
34            return super.onTouchEvent(event);
35        }
```

说明：

- 第 1 行：响应触摸事件的方法 onTouchEvent()，就可以实现鼠标单击和触摸功能。
- 第 3~8 行：把游戏的开始设置成直接触摸屏幕即可开始。
- 第 10、11 行：获得触屏的 X 和 Y 坐标值。
- 第 13 行：如果上下移动，判断蛇下一步的移动方向是左还是右。
- 第 16 行：更新向左移动。
- 第 20 行：更新向右移动。
- 第 23 行：如果左右移动，判断蛇下一步的移动方向是上还是下。
- 第 27 行：更新向上移动。
- 第 31 行：更新向下移动。

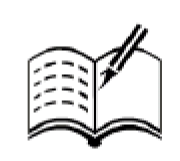
12.3.2　添加游戏的背景图片

因为游戏整体是通过 Bitmap 绘制的方法，通过点阵像素来完成游戏的全部运行显示和制作的过程，所以背景不能通过直接绘制图片的方法添加到游戏中，而是通过 Bitmap 绘制的方法将图片变为像素点，然后绘制到 Bitmap 上，最后将 Bitmap 添加到游戏的背景上，即借助于 BitmapFactory 获取位图，通过 Canvas 类的 drawBitmap()方法显示位图，从而实现背景的添加，这样就能达到美化游戏界面的效果了。代码如下：

```
public void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        Bitmap mpicture = BitmapFactory.decodeResource(this.getResources(),
                drawable.mypicture);
        RectF rectf = new RectF(0,0,480,800);
```

```
        canvas.drawBitmap(mpicture,null,rectf,null);
    …
}
```


Note

说明：

实现时需要在 res 目录下的 drawable 文件夹下添加图片文件 mypicture.png，实现效果如图 12-14 所示。

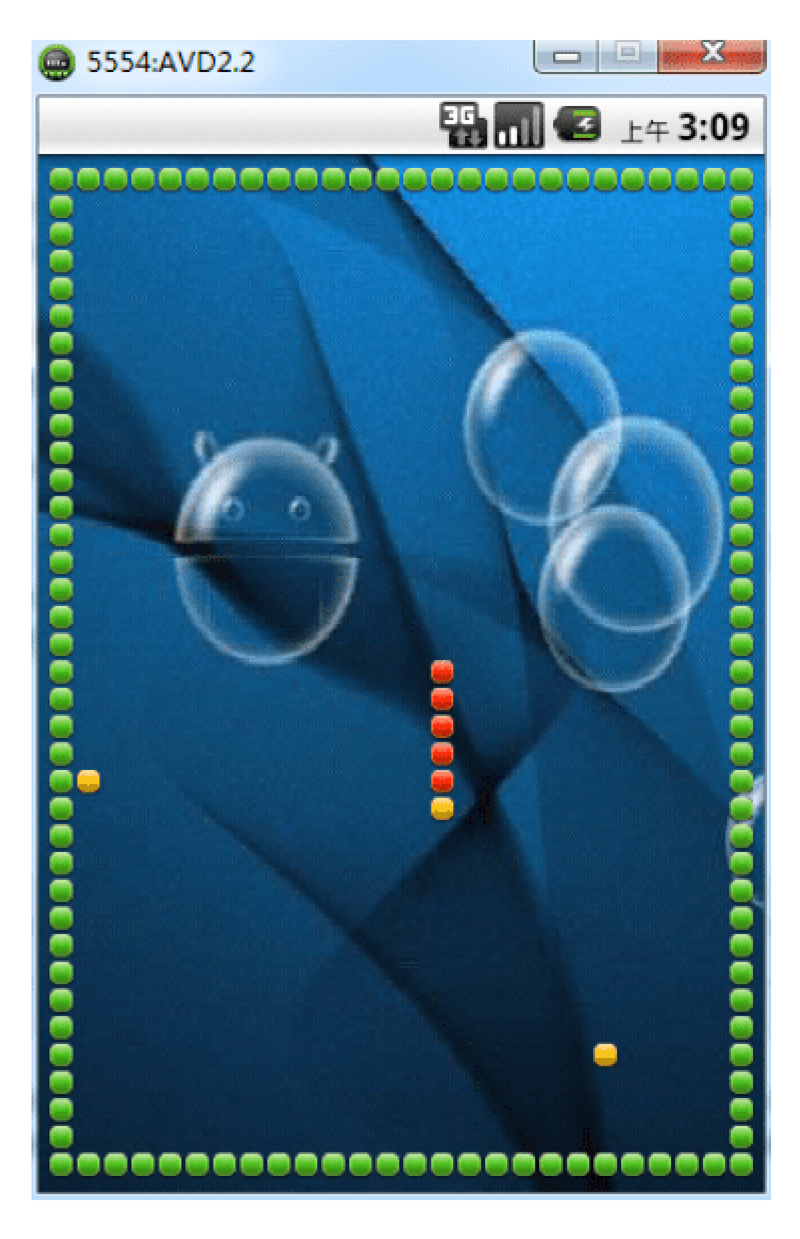


图 12-14　添加背景图片的实现效果

12.3.3　增加游戏的背景音乐

最后，给比较单调的游戏添加一个“金蛇狂舞”的背景音乐。游戏开始时，背景音乐就响起，直到游戏结束，背景音乐会一直循环播放，增加游戏乐趣。

（1）在 res 目录下新建一个 raw 文件夹，把背景音乐文件 goldensnake.mp3 放入。

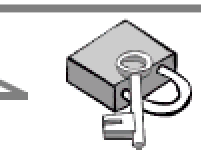
（2）增加一个 Music 类的内容，代码如下：

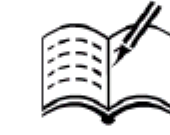
```
package com.example.android.snake;

    public class Music {
        private static MediaPlayer mp = null;

        public static void play(Context context, int resource) {
            stop(context);
            mp = MediaPlayer.create(context, resource);
            mp.setLooping(true);
            mp.start();
        }

        public static void stop(Context context) {
```





```
14                  if (mp != null) {
15                      mp.stop();
16                      mp.release();
17                      mp = null;
18                  }
19              }
20      }
```

说明：

- 第 4 行：声明一个音乐播放器。
- 第 8 行：实例音乐播放器。
- 第 9 行：设置循环播放。
- 第 10 行：设置音乐播放器开始播放。
- 第 13 行：设置音乐播放器停止播放。

（3）在 Snake.java 文件中添加如下代码（粗体代码）：

```
1   public void onCreate(Bundle savedInstanceState) {
2           …
3           if (savedInstanceState == null) {
4                   Music.play(this, R.raw.goldensnake);        //添加代码
5                   …
6       }
7       protected void onPause() {
8               Music.stop(this);                            //添加代码
9           …
10      }
```

说明：

- 第 4 行：播放背景音乐。R.raw.goldensnake 是资源文件，MP3 格式。
- 第 8 行：停止播放背景音乐。

这样，在运行游戏时就能听到背景音乐了。

12.4 本 章 小 结

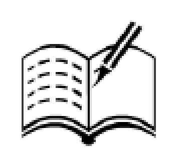
本章通过贪吃蛇的实例，介绍 Android 的图形绘制、贴图方法和游戏开发的基本逻辑和设计流程，同时对游戏加以扩充，增加触摸功能、背景图片和背景音乐效果，使其更具实用性。

选择以贪吃蛇为例作为切入点，有如下几个原因：

（1）贪吃蛇是手机上的一个有趣的游戏，实现简单却又极具可玩性。

（2）贪吃蛇可以有很多版本和变种，可以做得很简单，也可以做得很复杂，如可以设置多个关卡；可以是 2D，也可以是 3D。

（3）读者可以该实例为基础进行扩展，开发出更炫、更有趣的新版本的贪吃蛇游戏。


Note

12.5 习　　题

1．游戏开发中，监听屏幕触摸事件需重写 View 类的什么方法?

2．简要说明 Drawable、Bitmap、Canvas 和 Paint 的区别。

3．简要说明 RefreshHandle 类在游戏中的作用。

参考文献

[1] 吴亚峰，索依娜．Android 核心技术与实例详解．北京：电子工业出版社，2010

[2] 杨丰盛．Android 应用开发解密．北京：机械工业出版社，2010

[3] 张波，高朝琴，杨越．Google Android 解密．北京：人民邮电出版社，2010

[4] 李刚．疯狂 Android 讲义．北京：电子工业出版社，2011